Notes on the `earth` package

Stephen Milborrow

June 1, 2021

Contents

1 Introduction

The `earth` R package [19, 22] builds regression models using the techniques in Friedman's papers "Multivariate Adaptive Regression Splines" [7] and "Fast MARS" [8]. The package can be downloaded from `https://CRAN.R-project.org/package=earth`.

The term "MARS" is trademarked and thus not used in the name of the package. A backronym for "earth" is "Enhanced Adaptive Regression Through Hinges".

This document is a set of notes that accompanies the package. It can also be downloaded from `http://www.milbo.org/doc/earth-notes.pdf`.

The other vignette that comes with the package is "Variance models in earth" [17], which describes how to build variance models and generate prediction intervals for earth models. It can be downloaded from `http://www.milbo.org/doc/earth-varmod.pdf`.

Most users will find it unnecessary to read this entire document. Just read the parts you need and skim the rest. Much of this text was originally written in response to email from users.

2 Overview

Earth has numerous arguments, but many users will find that the following are all they need:

`formula, data`	Familiar from `lm`.
`x, y`	Alternative to the formula interface.
`degree`	The maximum degree of interaction. Default is `1`, use `2` for first-order interactions of the hinge functions.
`nk`	The maximum number of MARS terms. The default is determined semi-automatically from the number of predictors in `x`, but may need adjusting.
`trace`	Trace operation.

It's usually best not to subvert the standard MARS algorithm by toying with tuning parameters such as `thresh`, `penalty`, and `endspan`. Remember that we aren't seeking a model that best fits the training data, but rather a model that best fits the underlying distribution from which the data is drawn. Knowledgeable users may ignore this advice.

2.1 References

The Wikipedia article [24] is recommended for an elementary introduction to MARS `http://en.wikipedia.org/wiki/Multivariate_adaptive_regression_splines`.

The primary references are the Friedman MARS papers [7, 8]. Readers may find the MARS section in Hastie, Tibshirani, and Friedman [12] a more accessible introduction.

Faraway [5] 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`, and check out the book's errata.)

Friedman and Silverman [9] is recommended background reading for the MARS paper.

Earth's backward pass uses code from the `leaps` package [16] which is based on techniques in Miller [20].

If you use `earth` in a published document, please do the right thing and cite it (FAQ 13.2).

2.2 Other implementations

Given the same data, earth models are similar to but not identical to models built by other MARS implementations.

The differences stem from the forward pass where small implementation differences (or perturbations of the input data) can cause somewhat different selection of terms and knots, although similar GRSq's. These differences can even occur with the same source code compiled on different architectures.

The backward passes ususually give identical or near identical results, given the same forward-pass results.

The source code of earth is derived from the function `mars` in the `mda` package written by Trevor Hastie and Robert Tibshirani [13]. See also the function `mars.to.earth` (in the earth package).

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

StatSoft have an implementation which they call "MARSplines" `http://www.statsoft.com/textbook/stmars.html`.

SAS have an implementation which they call "ADAPTIVEREG" `https://support.sas.com/rnd/app/stat/procedures/adaptivereg.html`.

Most other implementations appear to be suitable only for smaller problems because they don't use Friedman's MARS fast update algorithm (equation 52 in the MARS paper).

2.3 Limitations

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

(i) Piecewise cubic models (to smooth out sharpness at the hinges).
(ii) Model slicing (`plotmo` goes part way).
(iii) Handling missing values.
(iv) Automatic grouping of categorical predictors into subsets.
(v) The h parameter of Fast MARS.

2.4 The forward pass

Understanding the details of the forward and backward passes will help you understand earth's return value and the admittedly large number of arguments. Figure 1 is an overview.

See Chapter 3 for a discussion of the termination conditions for the forward pass. Set `trace=2` or greater to trace the forward pass.

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` returned by the forward pass includes all terms before trimming back to `selected.terms`).

The `bx` matrix has a row for every observation (i.e. for every row in `x`). It has a column for each basis function (also referred to as a MARS term).

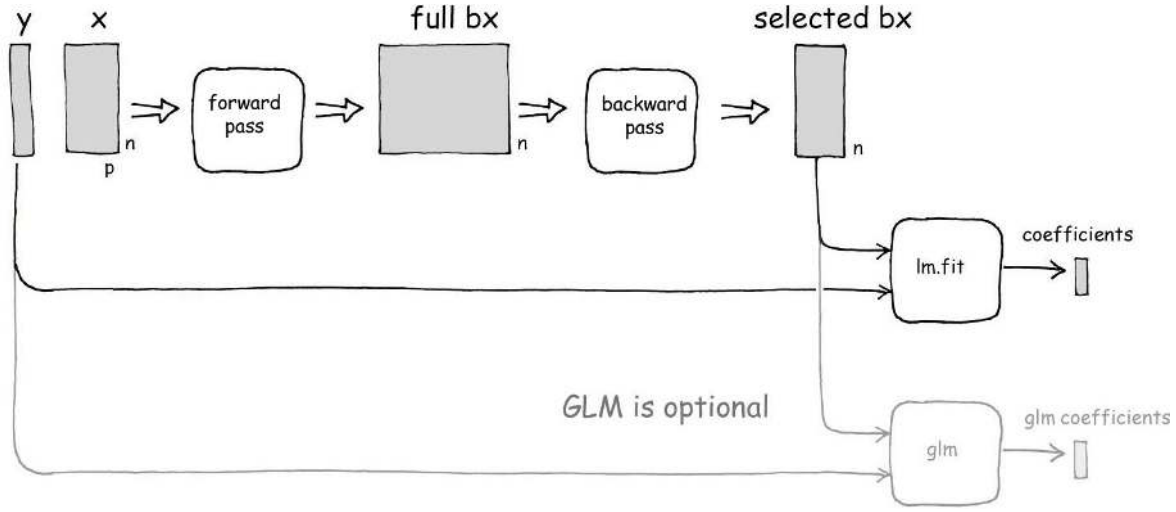

Figure 1: *Overview of* `earth`*'s internals*

An example `bx`:

```
      (Intercept)   h(x1-58)   h(x2-89)   h(89-x2)   h(56-x3)*h(x1-58)  ...
[1,]           1        3.2          0         56                   0  ...
[2,]           1        8.1          0         55                   0  ...
[3,]           1        3.7          0         54                   0  ...
....
```

2.5 The backward pass

The backward pass[1] is handed the set of terms `bx` generated by the forward pass. This full set of terms typically overfits the data. The job of the backward pass is find the subset of these terms that gives the best GCV (FAQ 13.6).

The backward pass applies a stepwise term deletion procedure: at each step it deletes a `bx` term to generate a smaller submodel. It starts with the full model returned by the forward pass, then at each step it deletes the term that gives the submodel with the lowest RSS (residual sum-of-squares). It saves the RSS, GCV, and set of terms for each such submodel in the vectors `rss.per.subset`, `gcv.per.subset`, and `prune.terms` (these fields are returned by earth). This process continues until only one term remains (the intercept term). The backward pass then selects the final model—the submodel with the best GCV. It updates `bx` to retain only the terms for this model.

After the backward pass, earth runs `lm.fit` to determine the `fitted.values`, `residuals`, and `coefficients`, by regressing the response `y` on the new `bx`. This is an ordinary least-squares regression of the response `y` on `bx` (Figure 1). If `y` has multiple columns then `lm.fit` is called for each column.

If a `glm` argument is passed to earth (Chapter 4), earth runs `glm` on (each column of) `y` in addition to the above call to `lm.fit`.

Set `trace=3` or greater to trace the backward pass.

[1]More correctly called the *pruning pass*, because backward stepping is only one of the pruning options available in earth. See earth's `pmethod` argument; `"backward"` is the default.

2.6 Execution time

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

(i) decreasing `degree` (because there are fewer combinations of terms to consider),

(ii) decreasing `nk` (because there are fewer forward pass terms),

(iii) increasing `minspan` (because fewer knots need to be considered),

(iv) decreasing `fast.k` (because there are fewer potential parents to consider at each forward step),

(v) increasing `thresh` (faster if there are fewer forward pass terms).

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`.

2.7 Model sizes and memory use

Earth doesn't impose specific limits on the model size. Total model size is limited only by the amount of memory on your system and your patience while waiting for the model to be built. For big models, earth automatically does memory housekeeping internally and invokes `gc`. Use `trace=1.5` to trace earth's memory usage.

The `x,y` interface to earth uses a little less memory than the formula interface, if all elements of the `x` and `y` matrices are of type `double`. This avoids an internal conversion of the matrices.

Reducing `nk` can significantly decrease memory (because it decreases the size of the basis matrix `bx` and of the various versions of it needed internally during model building).

Increasing `degree` doesn't change the memory requirements (but does increase running time).

The test suite file `earth/inst/slowtests/test.big.R` tests an earth model with 8 million cases and 100 variables, and a model with 80 million cases and 2 variables. The models were tested on a 64 bit Windows system with 32 gig of RAM. The models are built with earth's default arguments. Substantially bigger models are possible if `nk` is reduced or more physical memory is added to the machine.

Accumulated numerical error could potentially be a problem with a very large `x` matrix, although the very big models in the test suite seem unaffected by numerical error.

Building a big model can require more memory than physical memory. Memory page thrashing is often not as bad as one might expect, because during the search for knots the C code works with only a subset of the columns at a time.

2.8 Standard model functions

Standard model functions such as `case.names` are provided for earth objects and aren't explicitly documented. Many of these give warnings when the results aren't what you may expect. Pass `warn=FALSE` to these functions to turn of just these warnings. The full list of earth methods is:

`anova.earth`,
`case.names.earth`,
`coef.earth`,
`deviance.earth`,
`effects.earth`,
`extractAIC.earth`,
`family.earth`,
`fitted.earth`,
`fitted.values.earth`,
`hatvalues.earth`,
`model.matrix.earth`,
`plot.earth`,
`print.earth`,
`print.summary.earth`,
`resid.earth`,
`residuals.earth`,
`summary.earth`,
`update.earth`,
`variable.names.earth`,
`weights.earth`.

2.9 Multiple response models

If the response `y` has `k` columns then earth builds `k` simultaneous models.[2] 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 summed across all `k` responses during the forward and backward passes. Earth minimizes the overall GCV (the sum of the GCVs).

Once you have built your model, you can use `plotmo` [18] and its `nresponse` argument to see how each response varies with the predictors.

Here are a couple of (artificial) examples to show some of the ways multiple responses can be specified. Note that in R `data.frame`s unfortunately can't be used on the left side of a formula, so `cbind` is used in the first example. The last example uses the standard technique of specifying a tag (like `log.O3=`) to name a column.

[2]Note that this will be the case when a multilevel factor response is expanded by earth to multiple indicator columns, see Chapter 5 "Factors (categorical variables)".

Examples:

```
earth(cbind(O3,wind) ~ ., data=ozone1)    # formula interface

earth(O3 + wind ~ ., data=ozone1)         # use + on left of formula
                                          # requires earth version 5.0.0 (March 2019)

earth(ozone1[,-c(1,3)], ozone1[,c(1,3)])  # x,y interface

earth(x=data.frame(x1, x2, log.x3=log(x3)), y=data.frame(y1, y2))
```

Support for a plus sign `+` on the left of the formula was added in earth version 5.0.0 (March 2019)[3]. When using a plus this way, it's best to keep the left side of the formula simple, especially if there is a dot on the right.

Since earth attempts to optimize the set of basis functions for all models simultaneously, the results for each model won't be as "good" as building the models independently (where goodness is measured by GRSq). 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 the usual automatic scaling of `y` (via the `Scale.y` argument) doesn't 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.

For more details on using residual errors averaged over multiple responses see for example Section 4.1 of the FDA paper (Hastie, Tibshirani, and Buja [11]).

2.10 Weights

Case weights are implemented in earth using the standard technique of internally replacing the regression of `y` on `x` with a regression of `sqrt(weights) * y` on `sqrt(weights) * x`. This is invisible to the user – it happens internally in the earth function.

In the earth code, zero weights are internally converted to very small values (whereas in the `lm` code, cases with zero weights are removed; this is more rigorous but adds complexity when updating or plotting the model).

Weights are ignored when calculating and applying `minspan` and `endspan`—that is, for the purposes of determining the minimum spacing of knots, an observation is treated as a single observation regardless of its weight.

[3]Previous versions of earth treated a `+` on the left as arithmetic addition, as `lm` and `glm` still do.

A note on speed. Critical to the MARS fast update formula (equation 52 in the MARS paper) is the computational efficiency that results from processing the predictor values in order. However, multiplying by `sqrt(weights)` changes the order of the values, in general, so we can't (easily) use the formula after weights have been applied.[4] So instead of using the formula, the current implementation of weights does a full regression at each knot. This is slow. It could surely be optimized, but not to the extent of the code for unweighted observations.

2.11 Migrating from mda::mars

The classic `mda` package [13] has an implementation of MARS which predates earth.

Changing your code to use earth instead of a `mars` from the `mda` package 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. For details, see the documentation of the function `mars.to.earth` in the earth package.

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`".

A note on `wp` argument. Earth's internal normalization of `wp` is different from `mars`. Earth uses `wp <- sqrt(wp/mean(wp))` and `mars` uses `wp <- sqrt(wp/sum(wp))`. Thus in earth, a `wp` with all elements equal is equivalent to no `wp`. For models built with `wp`, multiply the GCV calculated by `mars` by `length(wp)` to compare it to earth's GCV.

Earth is faster than `mda::mars` for large models. This is primarily because earth's C code (i) re-arranges the data for better cache use in modern processors, (ii) uses the BLAS routines, (iii) uses Friedman's fast MARS techniques [8], and (iv) saves certain regression coefficients for re-use (see earth's `Use.beta.cache` argument).

[4] Consider for instance the hinge function `max(x - 123, 0)`. The `x` values greater than `123` that support the hinge function may be reordered after multiplication by `sqrt(weights)`; some of them becoming less than the weighted `123` and some remaining greater.

3 Termination conditions for the forward pass

The forward pass adds terms in hinge pairs until any of the following conditions is met.

(i) Reached the maximum number of terms `nk`.

(ii) Adding a term changes R^2 by less than 0.001.

(iii) Reached a R^2 of 0.999 or more.

(iv) GRSq is less than -10 (a pathologically bad GRSq, FAQs 13.11 and 13.12).

(v) No new term increases R^2 (possibly because we have reached numerical accuracy limits).

Not all the conditions above are strictly necessary (earth could just stop when it reaches `nk` terms). However, the additional conditions save time by terminating when it is pointless to continue, and also minimize the generation of terms with arbitrary knots due to numerical noise.

The default maximum number of terms `nk` is somewhat arbitrary. Thus if `print.earth` shows that the termination condition for our model is `Reached maximum number of terms`, we might consider increasing `nk`. Perhaps the forward pass was terminated too soon.

See also the FAQs
13.13 "How is the default number of terms `nk` calculated?", and
13.14 "Why do I get fewer terms than nk, even with pmethod=none?".

Set `trace=2` or greater to get details on the forward pass.

Note that GCVs (via GRSq) are used during the forward pass only as one of the (more unusual) stopping conditions. Changing the `penalty` argument doesn't change the knot positions.

For knowledgeable users only: The numbers 0.001 and 0.999 above can be changed by changing earth's `thresh` argument. Setting `thresh` all the way to zero (`thresh=0`) disables all termination conditions, except `nk` and conditions involving numerical limits. A potential problem is that this allows earth to continue processing even if numerical issues cause instability.

4 Generalized Linear Models (classification models)

Generalized Linear Models (GLMs) are a statistical technique often used when the response is binary, categorical, or a count. Earth builds a GLM if you use its `glm` argument. This earth-glm differs from a conventional GLM in that variables enter the model with hinges, instead of directly as linear variables. And earth will do variable selection, discarding variables that don't contribute. The disadvantage is that model statistics are no longer available (like standard errors on the coefficients).

To build a GLM model, earth first internally builds a conventional earth model, and then invokes the standard R function `glm` on the earth 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 backward pass. (See Figure 1 and Section 2.5 "The backward pass".) 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 extra fields in earth's return value: `glm.list`, `glm.coefficients`, and `glm.stats`.

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(earth.model)` as usual to see the model. Use `summary(earth.model, details=TRUE)` to see more details, but note that the printed p values for the GLM coefficients are meaningless (FAQ 13.8).

Use `plot(earth.model$glm.list[[1]])` to invoke `plot.glm` to plot the internal `glm` model.

The approach used for GLMs in earth was motivated by work done by Jane Elith and John Leathwick ([15] is a representative paper).

4.1 GLM examples

The examples below show how to specify earth-glm models. The examples are only to illustrate the syntax and not necessarily useful models. In these examples we use `trace=1` so earth shows how it expands the input data (as explained in Chapter 5 "Factors (categorical variables)" and Section 4.4 "Binomial pairs").

(i) **Two-level factor or logical response (binary response).**

Internally in earth, the response is converted to a single column of 1s and 0s.

```
binary.mod <- earth(survived~., data=etitanic,
                    glm=list(family=binomial), trace=1)
```

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

We mention that the function `plotd` can be useful for these responses (Section 9.4).

(ii) **Factor response (multinomial response).**

This example is for a factor with more than two levels. (For factors with just two levels, see the previous example.)

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

Internally in earth, the factor `pclass` is expanded to three indicator columns.[5] Section 5.2 has more detail. Because of the "masking problem", we mention that you might consider FDA for factor responses with more than two levels (Chapter 8).

(iii) **Binomial model specified with a column pair.**

This is a single response model, but specified with a pair of columns (Section 4.4 "Binomial pairs").

```
ldose <- rep(0:5, 2) - 2 # Venables and Ripley 4th edition page 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)
numalive <-  20 - numdead
pair <- cbind(numalive, numdead)
pairmod <- earth(pair ~ sex + ldose, glm=list(family=binomial), trace=1)
```

We can also specify the two columns using a plus sign `+` on the left of the formula:

```
pairmod2 <- earth(numalive + numdead ~ sex + ldose,
                  glm=list(family=binomial), trace=1)
```

The use of plus like this on the left of the formula requires earth version 5.0.0 or later (March 2019).[6]

The following illustrates use of a `probit` link, and (unnecessarily) increases `maxit`.

```
pairmod3 <- earth(numalive + numdead ~ sex + ldose,
                  glm=list(family=binomial(link=probit), maxit=100), trace=1)
```

[5]This differs from what would happen if we called `glm` directly with a multi-level factor response, where `pclass` would be treated as binary: the first level versus all other levels.

[6]The `lm` and `glm` functions don't work this way—they treat a `+` on the left of the formula as an addition—they arithmetically add the columns to yield a single column response.

(iv) **Poisson model (count data)**

Using insurance data from the `MASS` package:

```
library(MASS)
data(Insurance)
pois.mod <- earth(Claims ~ District + Group + Age + offset(log(Holders)),
                  data=Insurance, glm=list(family=poisson), trace=1)
```

With an `offset` term in the formula, the different methods used for calculating RSq by `earth` and `lm` become evident. Earth calculates RSq (and GRSq) *after* subtracting the offsets from the response and the fitted values.
This is a consequence of the fact that earth calculates RSq as `1-rss/tss`, whereas lm calculates RSq as `regression.sum.of.squares/tss`. Further discussion in FAQ 13.10 "Can R^2 be negative?".

(v) **Standard earth model**, the long way.

Using `family=gaussian` with an `identity` link builds a `glm` model which is equivalent to a standard earth model.

```
gauss.mod <- earth(numdead ~ sex + ldose,
                   glm=list(family=gaussian(link=identity)), trace=1,)

print(gauss.mod$coefficients == gauss.mod$glm.coefficients)  # all TRUE
```

4.2 GLM statistics printed by summary.earth

Earth prints some useful summary statistics for GLM models. For example, for the above Poisson model `summary(pois.mod)` includes:

```
nulldev df        dev df    devratio      AIC     iters     converged
 236.26 63     70.077 58        0.70      399         4             1
```

This shows some essential statistics such as the deviance with its degrees-of-freedom. Remember that these statistics are on the training data, so will tend to be optimistic about performance on independent data.

The `devratio` is the *deviance ratio* or "fraction of deviance explained", defined analogously to R^2 as

`devratio = 1 - deviance / null_deviance`.

The deviance-ratio is a number between 0 and 1, with values closer to 1 indicating a better fit. An intercept-only model has a deviance ratio of 0.

4.3 Weights with GLM models, and "non integer" warnings

If we pass weights to earth, those weights are used when building both the internal MARS model and the internal glm model.

The standard R `glm` function sometimes issues "non integer" warnings (whether `glm` is called directly or via earth). These warnings often can be safely ignored:
`https://stackoverflow.com/questions/12953045/warning-non-integer-successes-in-a-binom`
Sometimes using a weights argument can trigger a spurious non-integer warning from `glm`. Use `family=quasibinomial` instead of `family=binomial` to build the same model without the warning (the model coefficients will be the same but model statistics like the standard errors may differ).

4.4 Binomial pairs

Users of the `glm` function with binomial data will be familiar with the technique of specifying a binomial response as a two-column matrix, with columns for the number of successes and failures. Example (iii) on page 14 is an example. When the earth argument `glm=list(family=binomial)` is used, earth automatically detects when such columns are present in the response.[7]

Both earth and `glm` process these binomial pairs as follows. The two response columns are converted internally to a single column, the success fraction for each pair:
`nsuccess / (nsuccess + nfailure)`. This single column response is used to build the internal model.

By default, case weights are set to the response row sums (`nsuccess + nfailure` for each pair). If all the row sums are the same, this is equivalent to no weights. If weights are specified as an argument when invoking earth or `glm`, they get multiplied by these row sums. As a special case, if both `nsuccess` and `nfailure` are zero in a row, this is treated as a success fraction of zero with a zero weight.

The above method of generating a single-column response and weights for binomial pairs was introduced in earth version 5.0.0 (March 2019). It was introduced to make earth's handling of binomial pairs the same as `glm`. Previous versions of earth used the unweighted first column of the response to build the internal MARS model.

Issues with binomial pairs

With binomial pairs, some issues arise to do with how earth calculates the GCV and parameters like the `minspan`. These parameters are calculated using the number of observations, but the effective number of observations for a binomial pair is not well

[7]Since, unlike `glm`, earth supports multiple responses (Section 2.9), earth must determine if two-column response is a single binomial-pair response or a multiple response. Earth considers a response `y` to be a binomial-pair if all of the following are true
(i) the `family` is `binomial` or `quasibinomial`
(ii) `y` has two columns
(iii) all `y` values are non-negative integers
(iv) at least one row of `y` sums to greater than 1.
If `y` has two columns but these conditions aren't met, earth builds a multiple response model (Section 2.9).

defined. Is each row in the binomial pair matrix worth one observation, or worth the total number of successes and failures for that row? Probably each row is worth somewhere in between.

In practice, earth treats each row as a single observation (but weighted by the row sum). Since this may be less than the effective number of observations, earth may underfit the data (since it feels that there aren't enough observations to build a complex model). An intercept-only model is not uncommon.

To counteract underfitting, we can decrease the default `minspan` and `endspan` (using say `minspan=1` and `endspan=1`), and also consider reducing `penalty` (perhaps using `penalty=0` or `-1`). This risk here is that we may now *over*fit the data. There seems to be no unequivocally reliable approach. Use of the "long" form of the data could be considered (see the next section). Cross-validation may be of use here, if there is enough data.

Short versus long binomial data

Use the function `expand.bpairs` to convert the "short" form of the data (with a two-column binomial pair response) to the equivalent "long" form (with a single response column of TRUEs and FALSEs). See the help page of `expand.bpairs` for an example.

Models built with the short and long forms of the data won't be the same in general. If just `glm` is used (without earth), then the model coefficients and standard errors will be the same, but the residuals and model statistics like the deviance and AIC will differ. In an earth-glm model, in addition to these differences, the model coefficients will also differ (because in general earth generates a different set of hinges for the short and long data).

Here are example models built with short and long forms of data. The `earth.short.lin` and `earth.long.lin` models are more for illustration than actual use.

```
ldose    <- rep(0:5, 2) - 2 # Venables and Ripley 4th edition page 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)
numalive <- 20 - numdead

glm.short       <- glm(cbind(numalive,numdead) ~ ldose + sex, family=binomial)

earth.short     <- earth(cbind(numalive,numdead)  ~ ldose + sex,
                         glm=list(family=binomial))

earth.short.lin <- earth(cbind(numalive,numdead)  ~ ldose + sex,
                         glm=list(family=binomial),
                         # coerce earth to build a linear (no hinge) model with all vars
                         # (generated model matches the glm.short model above)
                         linpreds=TRUE, thresh=0, penalty=-1)

data.short <- data.frame(numalive, numdead, ldose, sex)

data.long <- expand.bpairs(data.short, c("numalive", "numdead"))
                         # data.long$numalive will be a fraction 0...1

glm.long       <- glm(numalive ~ ldose + sex, data=data.long, family=binomial)

earth.long     <- earth(numalive ~ ldose + sex, data=data.long,
                        glm=list(family=binomial))

earth.long.lin <- earth(numalive ~ ldose + sex, data=data.long,
                         glm=list(family=binomial),
                         linpreds=TRUE, thresh=0, penalty=-1)

coef(glm.short)
coef(earth.short.lin)    # same
coef(glm.long)           # same
coef(earth.long.lin)     # same
coef(earth.short)        # different if different hinge functions
coef(earth.long)         # different if different hinge functions
```

5 Factors (categorical variables)

This chapter explains how factors in the data get "expanded" before the matrices get passed to the MARS engine.

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

5.1 Factors in the predictors

Earth treats factors in the right side of the formula in the same way as standard R models such as `lm`. Thus factors are expanded using the current setting of `contrasts`. See the variable `sex` in the example in Section 5.3. This expansion happens whether earth is invoked via the formula or the x,y interface.

5.2 Factors in the response

Earth treats factors in the response in a non-standard way that makes use of earth's ability to handle multiple responses. This happens whether earth is invoked via the formula or the x,y interface.

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. For example, if the response is a factor with levels `"blue"`, `"green"`, and `"red"`, the response will be expanded to three columns like this (the actual data will vary but each row will have a single 1):

```
blue green red     # 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     1   0
   ...
```

This expansion to multiple columns (which only happen for factors with more than two levels) means that earth will build a multiple response model as described in Section 2.9 "Multiple responses".

Some details on factor expansion. For an earth factor response, the `contrasts` matrix is thus an identity matrix—see the help page of `contr.earth.response`. Earth response factors are handled in this way regardless of the global `options("contrasts")` setting, and regardless of whether the factor is ordered or unordered. In distinction, a standard treatment contrast on the *right* side of the formula has no first `"blue"` column, to prevent linear dependencies in the `x` matrix. See the help page of `contrasts` for details.

Note also that, in addition to factor responses, *paired binomial responses* are treated specially (Section 4.4 "Binomial pairs").

5.3 Factor example

Here is an example which uses the `etitanic` data to predict the passenger class. We use the optional `trace=1` here so earth shows the expanded factor names.

```
> data(etitanic)
> head(etitanic) # pclass and sex are 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[1046,5] with colnames survived sexmale age sibsp parch
y[1046,3] with colnames 1st 2nd 3rd
rest not shown here...
```

6 The `linpreds` argument

With the `linpreds` argument, we can specify which predictors should enter linearly, instead of in hinge functions.

We give a simple example where this might be useful. Starting with the standard earth model

```
fit1 <- earth(Volume ~ ., data = trees)
plotmo(fit1)
```

we see in the `plotmo` graphs (Figure 2, top row) or by running `evimp` that `Height` isn't as important as `Girth`. For collaborative visual evidence that `Girth` is a more reliable indicator of `Volume`, look at the last row of the following `pairs` plot (not shown):

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

Since we want the simplest model that describes the data, we may decide that `Height` should enter linearly, not in a hinge function (Figure 2, bottom row):

```
fit2 <- earth(Volume ~ ., data = trees, linpreds = "Height")
summary(fit2)
```

which yields

```
                coefficients
(Intercept)            4.221
Height                 0.343   # Height enters linearly
h(14.2-Girth)         -3.199
h(Girth-14.2)          6.405
```

In this example, the second simpler model has almost the same R^2 as the first model.

6.1 Specifying `linpreds`

We can make all predictors enter linearly like this (the single `TRUE` is recycled to the length of `linpreds`):

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

Other ways of specifying `linpreds`:

```
earth(Volume~., data=trees, linpreds=2)                    # column index in x
earth(Volume~., data=trees, linpreds=c("Height","Girth")) # multiple variables
```

Note that `grep` is used for matching when `linpreds` is a character vector. Thus `"wind"` will match all variables that have `"wind"` in their names. Use the regular expression `"^wind$"` to match only the variable named `"wind"`.

In the current implementation, the GCV penalty for predictors that enter linearly is the same as that for predictors with knots. One could argue that that isn't quite correct; linear terms should be penalized less. (It depends on whether we make an *a priori* decision to treat the variable linearly, or make the decision based on building a preliminary model from the same data.)

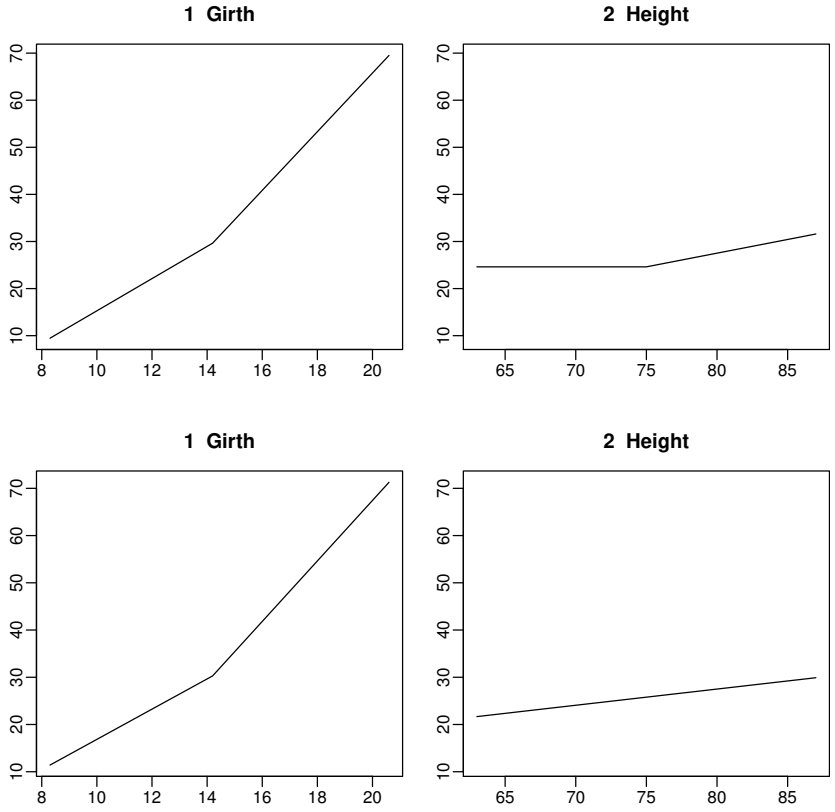

Figure 2: *The* `linpreds` *argument.*
Top row: Standard `earth` *model of the trees data.*
Bottom row: Same, but with ***`Height`*** *entering linearly (*`linpreds="Height"`*).*

6.2 Generating the same model as `lm`

Sometimes we would like to generate the same model as `lm`, with all predictors entering linearly. But the `linpreds` argument doesn't stipulate that a predictor *must* enter the model, only that if it enters it should enter linearly. If a variable has negligible additional predictive power, earth won't include it.

To circumvent this automatic variable selection, we can increase the likelihood (but not guarantee) that earth includes all variables by using both the following arguments:

(i) `thresh=0`. Tell the forward pass to include a predictor even if it has very little predictive power (Chapter 3, last paragraph). Note that linearly dependent variables may still be excluded.

(ii) `penalty=-1`. Tell the backward pass not to discard any terms (FAQ 13.15). Also prevents the forward pass from terminating because GRSq is too negative (Chapter 3 condition (iv)).

Example:

```
earth(Volume~., data=trees, linpreds=TRUE, thresh=0, penalty=-1)
```

6.3 Automatically treating a predictor as linear

Under certain conditions, earth will *automatically* enter a predictor linearly: During the forward pass, if earth discovers that the best knot is at the predictor minimum, then earth adds the predictor to the model linearly (instead of in a hinge function). The left side of Figure 3 illustrates. This means that there can be negative values in the MARS basis matrix `bx` (whereas in the classic MARS basis matrix, all values are non-negative).

The right side of Figure 3 illustrates the classic MARS behavior in this situation. Set the argument `Auto.linpreds=FALSE` to get this behavior. All values in the basis matrix will be non-negative.

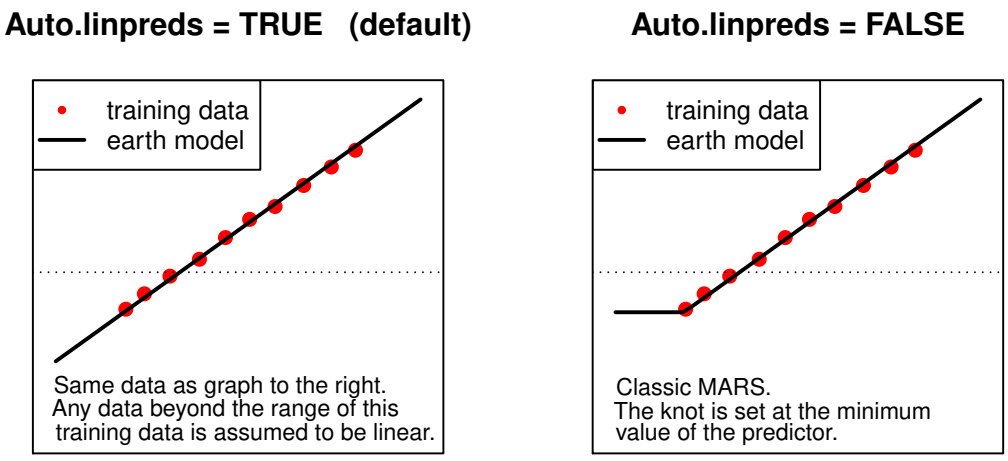

Figure 3: *The Auto.linpreds argument.*

7 The `allowed` argument

You can specify how variables are allowed to enter MARS terms with the `allowed` argument. Within each step of the forward pass, earth calls the `allowed` function after discovering the best knot for a variable. The potential term is considered for inclusion only if the `allowed` function returns `TRUE`. The default function always returns `TRUE`.

(Note added in 2018: In retrospect, the interface is probably too complicated. Instead of working through the following explanations, you may want to tweak the example code in Section 7.2 or look at examples on the CrossValidated web site.)

The `allowed` function should have the following arguments

```
function(degree, pred, parents, namesx, first)
```

where

`degree` is the interaction degree of the candidate term. Will be 1 for additive terms.

`pred` is the index of the candidate predictor. A predictor's index in `pred` is the column number in the input matrix `x` after factors have been expanded. Use earth's `trace=1` argument to see the column names after expansion.

`parents` is the candidate parent term's row in `dirs`.

`namesx` is optional and if present is the column names of `x` after factors have been expanded.

`first` is optional and if present is TRUE the first time your `allowed` function is invoked for the current model, and thereafter FALSE, i.e. it is TRUE once per invocation of `earth`.

7.1 Examples

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 isn't much use, because it's simpler to exclude the predictor from the input matrix when invoking earth:

```
a1a <- 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 we would want to single out humidity from interactions in the ozone data.)

```
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
}
a2 <- earth(O3 ~ ., data = ozone1, degree = 2, allowed = example2)
print(summary(a2))
```

Details on the above code: The `parents` argument is the candidate parent's row in the `dirs` matrix (`dirs` is described in the `Value` section of the `earth` help page). Each entry of `parents` is 0, 1, -1, or 2, and we index `parents` on the predictor index. Thus `parents[pred]` is non-zero if `pred` is in the parent term. (Yes, it's confusing. Maybe the easiest approach is to find the example in this chapter that is closest to what you want, and modify it for your needs.)

The following example allows only the specified predictors in interaction terms. Interactions are allowed only for predictors in `allowed.set`, which you can change to suit your needs.

```
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(allowed.set != pred) || any(parents[-allowed.set])))
        return(FALSE)
    TRUE
}
a3 <- earth(O3 ~ ., data = ozone1, degree = 2, allowed = example3)
print(summary(a3))
```

7.2 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 (Section 5.1). The first example above (the one that disallows `Height`) can be rewritten as

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

The next page has a few more examples.

```
#--- no predictor in PREDICTORS is allowed to interact with any predictor in PARENTS
#--- but all other interactions are allowed

PREDICTORS <- c("age")
PARENTS <- c("survived", "parch")

example4 <- function(degree, pred, parents, namesx)
{
    if (degree > 1) {
        predictor <- namesx[pred]
        parents   <- namesx[parents != 0]
        if((any(predictor %in% PREDICTORS) && any(parents %in% PARENTS)) ||
           (any(predictor %in% PARENTS)    && any(parents %in% PREDICTORS))) {
            return(FALSE)
        }
    }
    TRUE
}
a4 <- earth(sex~., data=etitanic, degree=2, allowed=example4)
plotmo(a4)

#--- predictors in PREDICTORS are allowed to interact with predictors in PARENTS
#--- but no other interactions are allowed

PREDICTORS <- c("age")
PARENTS    <- c("survived", "parch")

example5 <- function(degree, pred, parents, namesx)
{
    if (degree <= 1)
        return(TRUE)
    predictor <- namesx[pred]
    parents   <- namesx[parents != 0]
    if((any(predictor %in% PREDICTORS) && any(parents %in% PARENTS)) ||
       (any(predictor %in% PARENTS)    && any(parents %in% PREDICTORS))) {
        return(TRUE)
    }
    FALSE
}
a5 <- earth(sex~., data=etitanic, degree=2, allowed=example5)
plotmo(a5)
```

7.3 Further notes on the `allowed` argument

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.

8 Using earth with fda and mda

Earth can be used with the functions `fda` and `mda` in the `mda` package [13] to build Flexible Discriminant Analysis (FDA) and Multiple Discriminant Analysis (MDA) models.

You can pass arguments such as `degree=2` to earth by including them in the call to `fda`. Use the earth argument `keepxy=TRUE` if you want to call `plotmo` later. 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`'s help page).

Example (this gives the right side of Figure 4):

```
library(mda)
(fda <- fda(Species~., data=iris, keep.fitted=TRUE, method=earth, keepxy=TRUE))
summary(fda$fit)  # examine earth model embedded in fda model
plot(fda)         # right side of the figure
```

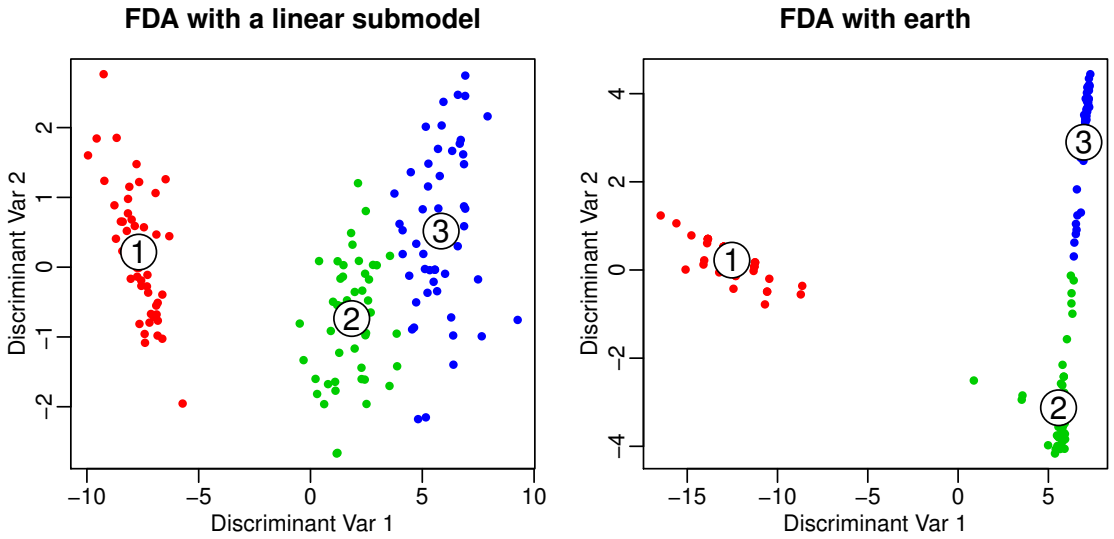

Figure 4: *Left: FDA of the iris data, built on a linear model.*
Right: FDA built with an `earth` *model using the code in the text. Note the better grouping of classes.*

The graphs show the training observations transformed into the discriminant space. This transformation is done by the regression function plugged into `fda` *and by optimal scoring (Section 8.1). There are three classes in this example so we have two discriminant variables. A new observation is classified by* `predict.fda` *as the class of the nearest centroid in discriminant space (the centroids are at the ringed numbers 1, 2, and 3).*

Using `plotmo` we can plot the per-predictor dependence of the `fda` variates like this:

```
plotmo(fda, type="variates", nresponse=1, clip=F) # 1st disc var (Figure 5)
plotmo(fda, type="variates", nresponse=2, clip=F) # 2nd disc var (not shown)
```

We can also look at the earth model embedded in the FDA model:

```
plotmo(fda$fit, nresponse=1, clip=F) # earth in FDA, 1st disc var (Figure 6)
plotmo(fda$fit, nresponse=2, clip=F) # earth in FDA, 2nd disc var (not shown)
```

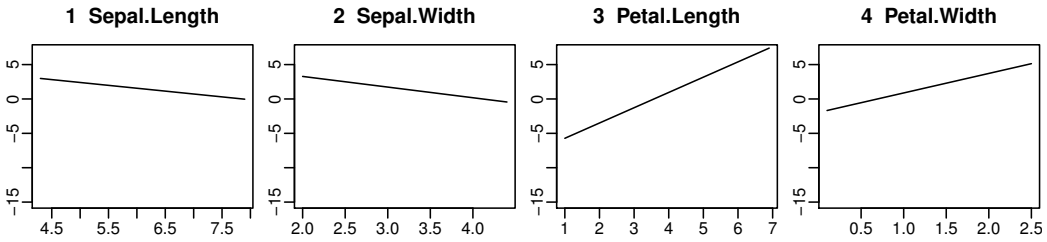

Figure 5: `plotmo` *graphs of the FDA model with a linear submodel (the left of Figure 3). The graphs show the contribution of each predictor to the first discriminant variable. The second discriminant variable isn't shown here.*
(The code to generate this plot isn't in the text.)

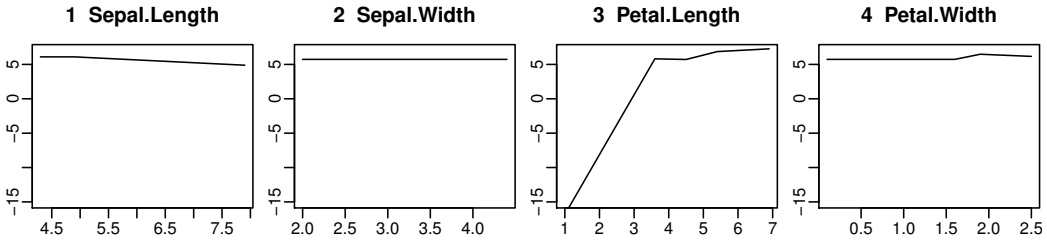

Figure 6: `plotmo` *graphs of the FDA model with an* `earth` *submodel (the right of Figure 4). The graphs show the contribution of each predictor to the first discriminant variable.*

Figure 7: `plotmo` *graphs of the* `earth` *model embedded in the FDA model (first discriminant variable before scoring). Earth didn't include* `Sepal.Width`.

8.1 A short introduction to Flexible Discriminant Analysis

Flexible Discriminant Analysis (FDA) is Linear Discriminant Analysis (LDA) on steroids. LDA uses a hyperplane to separate the classes. FDA replaces this hyperplane with a curved or bent surface to better separate the classes. The trick FDA uses to achieve this is to convert the classification problem into a regression problem. This allows us to plug in "any" regression function to generate the discriminant surface. If we plug in a linear regression function, FDA will generate a hyperplane, just like LDA. If we plug in `earth`, FDA will generate a surface defined by MARS hinge functions.

FDA converts a classification problem into a regression problem via *optimal scoring* (Figure 8). Essentially, this creates a new response variable by assigning new numbers (scores) to the factor levels in the original response. So for example `setosa=1`, `versicolor=2`, and `virginica=3` may become `setosa=1.2`, `versicolor=-1.2`, and `virginica=0`.

Actually, FDA creates several response variables like this, each with its own set of scores. If there are K response classes, FDA creates $K-1$ variables. So for the Iris dataset, which has three classes (or "levels" in R parlance), we have two discriminant variables (Figure 4). For a binary response FDA creates one discriminant variable, and the discriminant space is one dimensional. Note that the dimension of the discriminant space depends on the number of classes, not on the number of predictors — a nice example of dimensionality reduction. Sometimes the best prediction results on independent data are obtained if we use only some of the discriminant variables, and thus a further reduction in dimensionality is possible.

Further details may be found in Hastie et al. [12] Section 12.5 and the FDA paper (Hastie, Tibshirani, and Buja [11]).

Using FDA is usually recommended for a response that is a factor with more than two levels, rather than using a regression function like `lm` or `earth` directly on an indicator matrix. This is because of the "masking problem" (e.g. Hastie et al. [12] Section 4.2

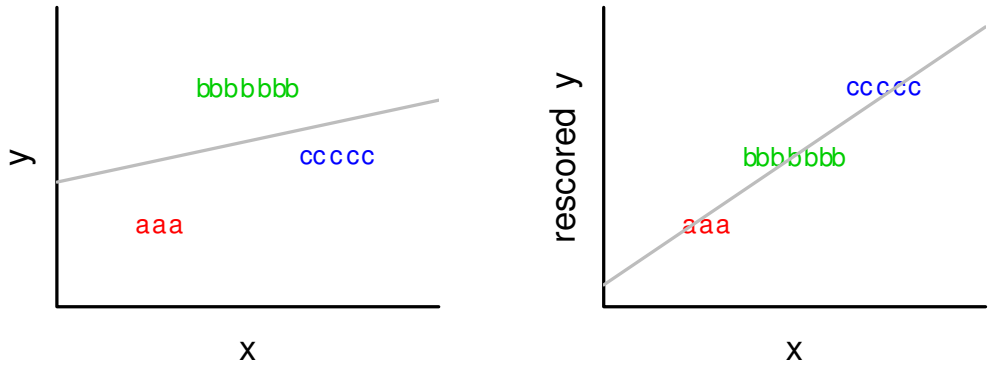

Figure 8: *Left: Some toy data with three response categories.*
Right: Rescoring assigns a new number to each category so the data are in a better form for linear separation.

"Linear Regression of an Indicator Matrix"). In practice the best advice is try it and see if you get better results on your data.

Two advantages of FDA are (i) FDA will often perform better than LDA (or QDA) because it generates a flexible surface to separate the classes, and (ii) for responses which have more than two levels, FDA will often perform better than regression on an indicator matrix because it doesn't suffer from the masking problem.

We mention that the acronym FDA for "Flexible Discriminant Analysis" is not to be confused with the same acronym for "Functional Data Analysis" [23].

TODO When is FDA exactly equivalent to LDA?

9 Plots

This chapter first describes the graphs produced by `plot.earth` then looks at a few other plots.

9.1 Short version of this chapter

For readers who don't wish to read this entire chapter, here is the least you need to know.

The `plot.earth` function produces four graphs (Figure 9).

Use the Model Selection plot to see how the fit depends on the number of predictors, how the final model was selected at the maximum GCV, and so on.

Use the Residuals vs Fitted graph to look for outliers and for any obviously strange behavior of the fitted function.

You can usually ignore the other two graphs.

9.2 Interpreting `plot.earth` graphs

The graphs plotted by `plot.earth`, apart from the Model Selection plot, are standard tools used in residual analysis and more information can be found in most linear regression textbooks. The `plot.earth` function is a wrapper around the `plotres` function in the `plotmo` package, so see also the documentation for that function.

Heteroscedasticity of the residuals isn't as important with earth models as it is with linear models, where homoscedasticity of the residuals is used a check that a linear model is appropriate. Also, in linear models homoscedasticity of the residuals is required for the usual linear model inferences (such as calculation of p values), which isn't done with earth models. You can model the variance of the earth model residuals using the `varmod` arguments as described in the "Variance models in earth" vignette.

9.2.1 Nomenclature

The *residuals* are the differences between the values predicted by the model and the corresponding response values. The *residual sum of squares* (RSS) is the sum of the squared values of the residuals.

R^2 (`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).[1]

The *Generalized Cross Validation* (GCV) is a form of the RSS penalized by the effective

[1]Not quite true, see FAQ 13.10 "Can R^2 be negative?"

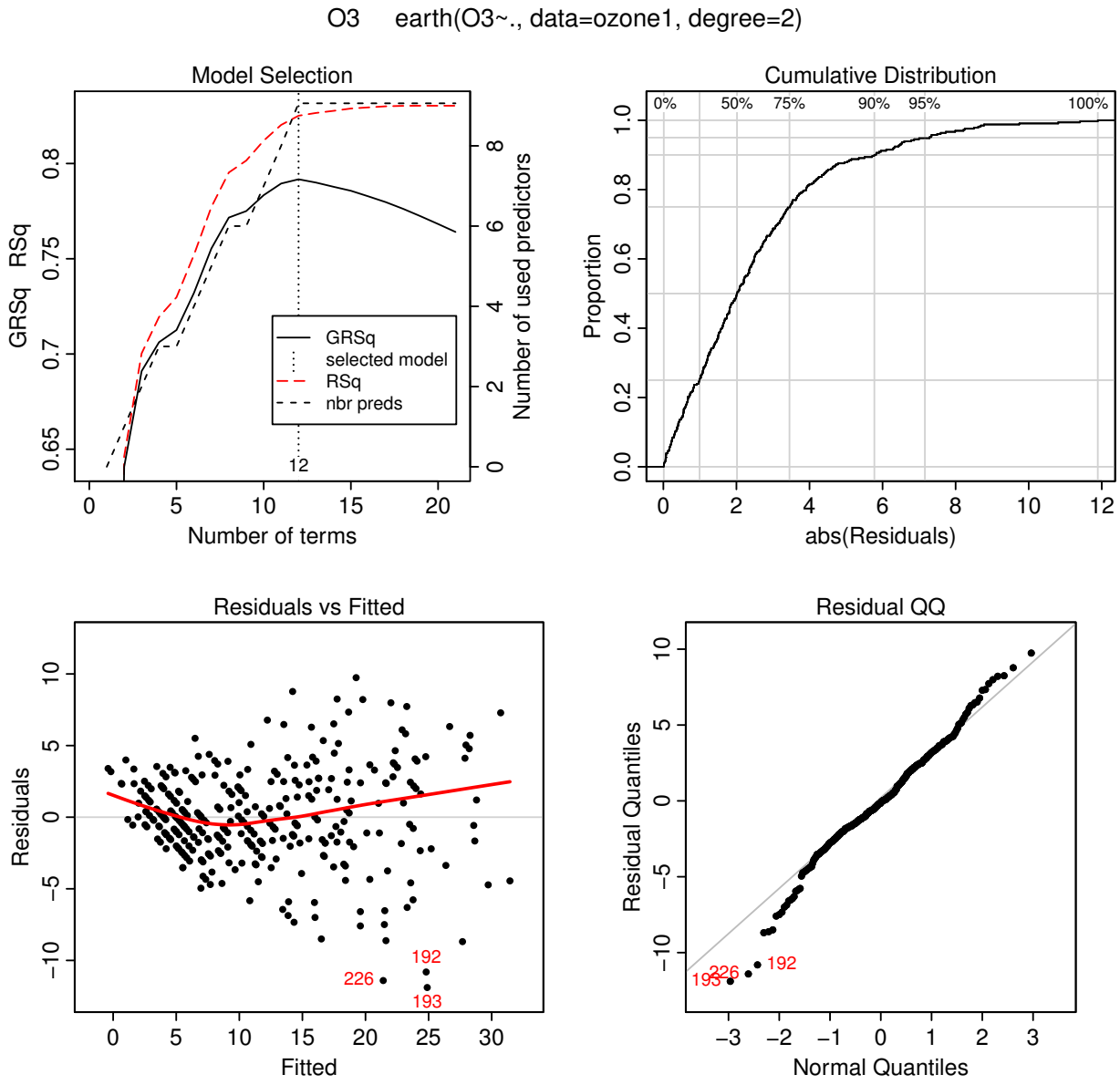

Figure 9: *Graphs produced by* `example(plot.earth)`.

number of model parameters (and divided by the number of observations). More details can be found in FAQs 13.6 and 13.7. The *GRSq* normalizes the GCV in the same way that the R^2 normalizes the RSS (see FAQ 13.11 and the definition of GRSq in the `Value` section of earth's help page).

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.

9.2.2 The Model Selection graph

For concreteness, the description of the graphs here is based on the plot produced by `example(plot.earth)` and shown in Figure 9. (Your version of earth might produce slightly different graphs.)

In the example Model Selection graph (top left of Figure 9) the RSq and GRSq lines 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 dotted line is positioned at the selected model. This will be at the maximum GRSq, unless `pmethod="none"` was used. In our example the vertical dotted line indicates that the best model has 12 terms and uses all 9 predictors (the number of predictors is shown by the black dashed line).

We can also see the number of predictors and terms we would need if we were prepared to accept a lower GRSq (you can use the earth parameter `nprune` to trim the model).

To reduce clutter and remove the right-hand axis, use `col.npreds=0`.

9.2.3 The Residuals vs Fitted graph

The Residuals vs Fitted graph (bottom left of Figure 9) 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.

The red line is a `lowess` fit. (Readers not familiar with `lowess` fits can think of them as fancy moving averages.) In this instance the mean residual diverges at low and high fitted values.

Ideally the residuals should show constant variance i.e. the residuals should remain evenly spread out, or homoscedastic, as the fitted values increase. (However, in earth, constant variance of the residuals isn't as important as it is in linear models.) In the example graph we see heteroscedasticity — 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 heteroscedasticity and possibly simplify the model, we can refit the model after performing a transform on the response. A cube root transform, for instance, evens out the residuals for this data (Figure 10, middle plot):

```
fit <- earth(O3^(1/3) ~ ., data = ozone1, degree = 2)
plot(fit)
```

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 — an expert may say that taking the cube root is meaningless, or conversely may say that it is essential).

(Be aware that R^2 and related statistics are affected by a transform on the response such as cube root we used above. If you are comparing models, for the model on the

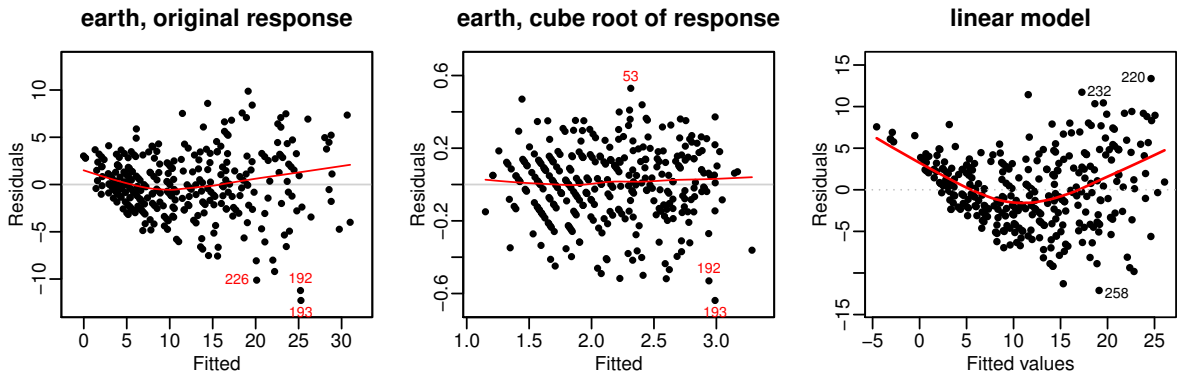

Figure 10: *Three models built from the ozone data.*
Left: The residuals of an earth model (same as bottom left of Figure 9).
Middle: The residuals of an earth model built on the cube root of the response.
Right: The residuals of a linear model.

transformed data, backtransform the predicted values by cubing them before evaluating the R^2.)

Compare the residuals of the earth model to the linear model (Figure 10, right plot), and notice how the red smooth lines show that the earth model is more successful at modeling non-linearities in the data. The code for the linear model plot is:

```
fit.lm <- lm(O3 ~ ., data = ozone1)
plot(fit.lm, which=1)    # which=1 for the residuals plot only
plotres(fit.lm, which=3) # alternative aproach using plotres (not shown)
```

One should always look at the residuals themselves as well as looking at the `lowess` fit, which is itself an approximation. However, in the example plot the `lowess` line appears reliable.

Cases 192, 193, and 226 have the largest residuals and seem to 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`. Conversely, increase `id.n` to label more residuals.) As a general rule, it is worthwhile investigating cases with large residuals. Perhaps they should be excluded when building the model. Conversely, it is possible that they reveal something important about the data that could warrant changes to the model. It may also be worthwhile to look at cases with *small* residuals if there is non-linearity in that region. To see the example input matrix ordered on the magnitude of the residuals, use `ozone1[order(abs(earth.model$residuals)),]`.

Sometimes groups of residuals appear in a series of parallel lines (e.g. Figure 10 middle plot). Usually these lines don't indicate a problem. They are formed when a set of plotted points has the same predicted or observed value, often due to discretization in the measurement of the observed response (e.g. by rounding to the nearest inch).

The "Variance models in earth" vignette [17] has more discussion on residual plots.

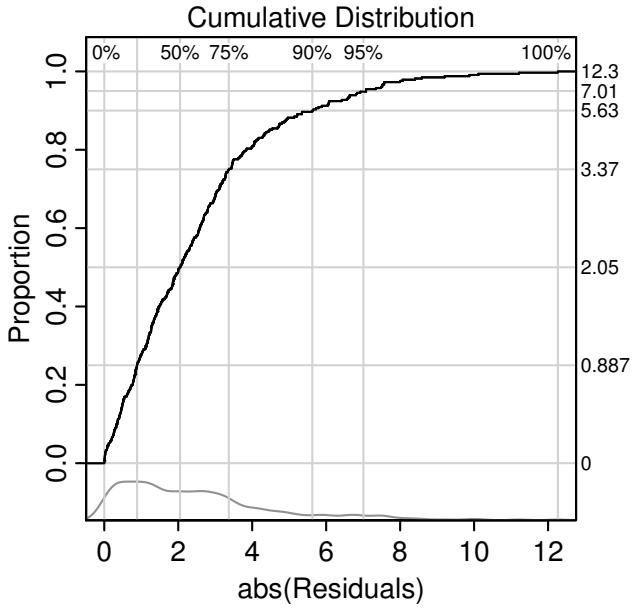

Figure 11:
Cumulative Distribution plot with `info=TRUE`

9.2.4 The Cumulative Distribution graph

The Cumulative Distribution graph (Figure 9 top right and Figure 11) 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.

This plots a step function for the n training cases: for each step $1/n$ in the vertical direction, in the horizontal direction we move to the next biggest absolute residual. An approximate bell-shaped distribution of absolute residuals will translate to an approximate S shape in the cumulative distribution plot (which is not really the situation in this example).

In the example graph, the median absolute residual is just over 2.0 (look at the vertical gray line for 50%). We see that 95% of the absolute values of residuals are less than about 7.0 (look at the vertical gray line for 95%). So in the training data, 95% of the time the predicted value is within 7.0 units of the observed value.

Pass `info=TRUE` to `plot.earth` to determine numbers like the 7.0 with more precision, as illustrated in Figure 11.

9.2.5 The QQ graph

The QQ (quantile-quantile) plot (bottom right of Figure 9) compares the distribution of the residuals to a normal distribution. If the residuals are distributed normally they will lie on the line. (Normality of the residuals often isn't too important for earth models, but the graph is useful for discovering outlying residuals and other anomalies.) 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 the 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.

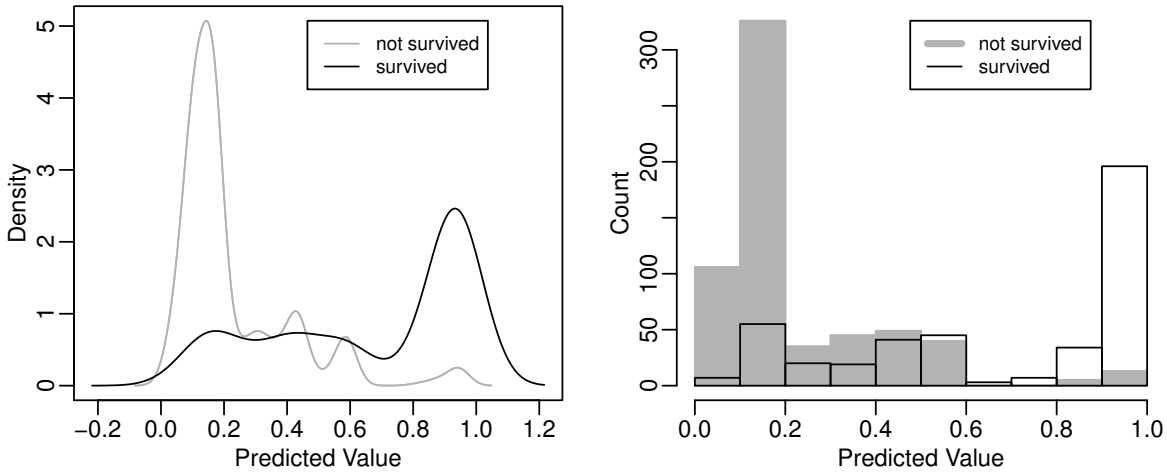

Figure 12: `plotd`

9.3 Earth-glm models and `plot.earth`

By default, the `plot.earth` function ignores the `glm` part of the model, if any.[8] The plotted residuals are residuals from earth's call to `lm.fit` after the backward pass, not `glm` residuals (although you change this using `plot.earth`'s `type` argument).

9.4 Earth-glm models and `plotd`

For earth-glm models, `plotd` (in the earth package) can be convenient. Example (Figure 12):

```
fit <- earth(survived ~ ., data=etitanic, degree=2, glm=list(family=binomial))

plotd(fit)                # left side of figure
plotd(fit, hist=TRUE)     # right side of figure
```

We can use `plot.glm` to plot the `glm` model within the earth model like this (not shown):

```
data(etitanic)
fit <- earth(survived~., data=etitanic, glm=list(family=binomial))
par(mfrow=c(2,2))       # four figures on one page
plot(fit$glm.list[[1]]) # invoke plot.glm
```

9.5 Cross-validated models and `plot.earth`

Earth builds cross-validated models with the `nfold` argument (Chapter 10 "Cross-Validation"). The Model Selection plot will show cross-validation statistics, but only if `keepxy=TRUE` was also used when building the model. (The cross-validation statistics are ignored in the other plots generated by `plot.earth`.) Figure 13 shows an example:

[8] Earth-glm models are models created with earth's `glm` argument, Chapter 4.

```
fit <- earth(survived ~ ., data = etitanic, degree=2, nfold=5, keepxy=TRUE)
plot(fit, which=1, col.rsq=0)  # which=1 for Model Selection plot only (optional)
```

In Figure 13, as usual the vertical black dotted line shows the optimum number of terms determined in the standard way at the peak GCV.

The pale pink lines show the out-of-fold RSq's for each fold model. The red line is the mean out-of-fold RSq for each model size.

The vertical red dotted line is at the maximum of the red line, i.e., the vertical line shows the optimum number of terms estimated by cross-validation. This is *CV-with-averaging*; another approach (not supported by earth) is *CV-with-voting* which uses the modal number-of-terms, i.e., the number-of-terms that is most often selected at a fold.

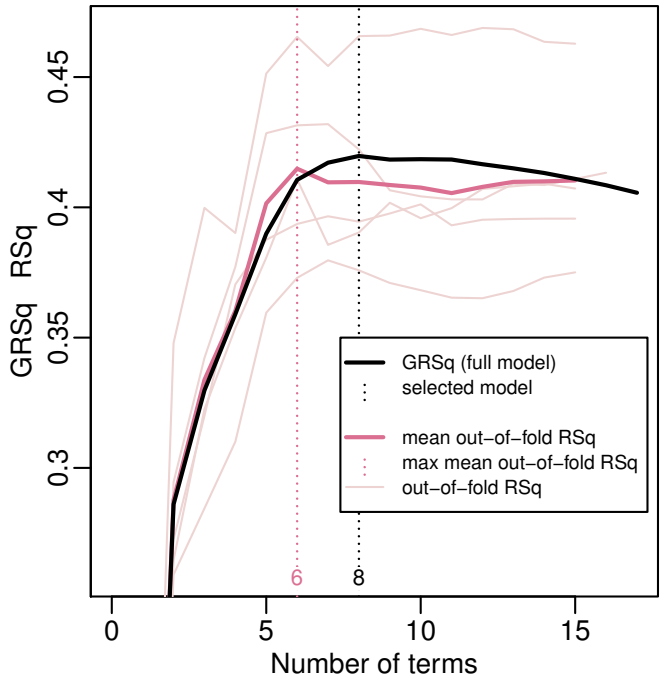

Figure 13: *A cross-validated* `earth` *model*

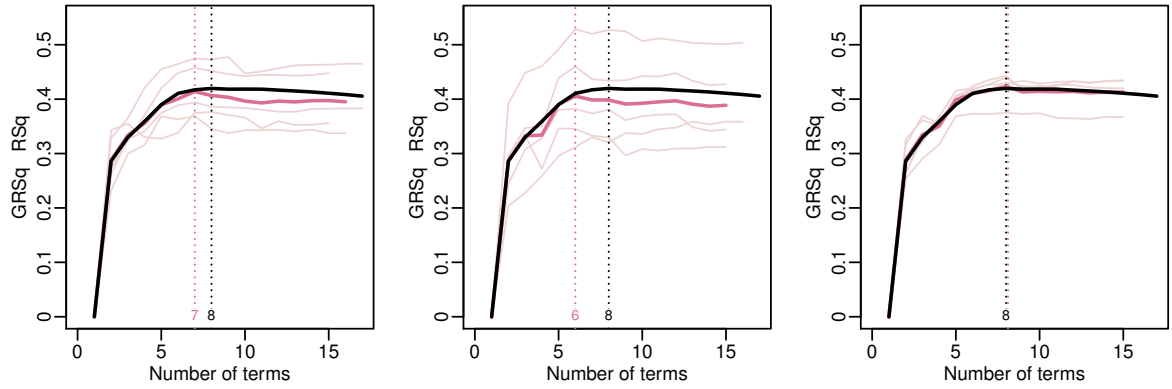

Figure 14: *The same model, cross-validated three times. Note the random variation in the cross-validation* `RSq`*s as* `earth` *partitions the data into folds differently for each run.*

Ideally the number of terms selected using GRSq would always match the number of terms determined by cross-validation, and the two vertical lines coincide (although `plot.earth` jitters the lines slightly to prevent overplotting). In reality, the vertical lines are usually close but not identical. In the following example, note how the graph varies as the cross-validation folds vary in each invocation of earth (Figure 14):

```
plot1 <- function()
{
    fit <- earth(survived~., data = etitanic, degree=2,
                 nfold=5, keepxy=TRUE)
    plot(fit, which=1, ylim=c(0, .5), col.rsq=NA, legend.pos=NA)
}
set.seed(1)
plot1()
set.seed(2)
plot1()
set.seed(3)
plot1()
```

The above code keeps the vertical axis range constant across all three graphs with `ylim=c(0,.5)`, reduces clutter with `col.rsq=NA`, and removes the legend with `legend.pos=NA`.

We mention that in Figures 13 and 14 the cross-validation results are consistent with the results obtained in the standard way using the GCV. The solid black and red lines are close. The vertical dotted red line dances around, but mostly because of the flatness of the curve after about 6 terms. The position of the vertical line would be more stable if we used the one-standard-error rule (not supported by earth) or the `ncross` argument.

For the curious, `plot.earth.models` can be used to compare the GCV curve of models built at each fold. Example (Figure 15):

```
fit <- earth(survived ~ ., data=etitanic, degree=2,
             nfold=3, keepxy=TRUE)
plot.earth.models(fit$cv.list, which=1, ylim=c(0, .5))
```

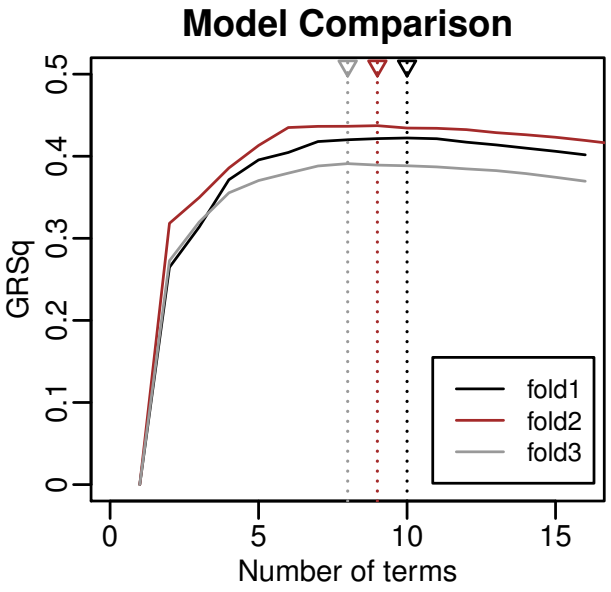

Figure 15: *Comparing the GRSq curves of models at three cross-validation folds.*

10 Cross-validating earth models

In this chapter we assume that you already know the basics of cross-validation (i.e., partition the data into `nfold` subsets, repeatedly build a model on all but one of those subsets, measure performance on the left-out data).

Use cross-validation to get an estimate of R^2 on independent data or to select a model. This is an alternative to the GRSq used by earth. (FAQ 13.6).

Here is an example. Note the `nfold` parameter and the cross-validation R^2, called `CVRSq` below. The left side of Figure 16 illustrates this model.

```
> fit <- earth(survived ~ ., data=etitanic, keepxy=TRUE, degree=2, nfold=5)
> summary(fit)

    ... usual text omitted here ...

GCV 0.14  RSS 139  GRSq 0.42  RSq 0.45  CVRSq 0.41

Note: the cross-validation sd's below are standard deviations across folds

Cross validation:   nterms 7.60 sd 0.84    nvars 5.40 sd 0.52

     CVRSq    sd     ClassRate    sd     MaxErr  sd
      0.41 0.064          0.79 0.032       -1.2  1
```

Cross-validation is done if `nfold` is greater than 1 (typically 5 or 10). Earth first builds a standard model with all the data as usual. This means that the standard fields in earth's return value appear as usual, and will be displayed as usual by `summary.earth`. Earth then builds `nfold` cross-validated models. For each fold it builds an earth model with the in-fold data (typically nine-tenths of the complete data) and using this model measures the R^2 from predictions made on the out-of-fold data (typically one-tenth of the complete data). The final mean `CVRsq` printed by `summary.earth` is the mean of these out-of-fold R^2s.

The cross-validation results go into extra fields in earth's return value. All of these have a `cv` prefix — see the `Value` section of the earth help page for details. For details on statistics like `ClassRate`, see Section 10.7 "Cross-validation statistics returned by earth".

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

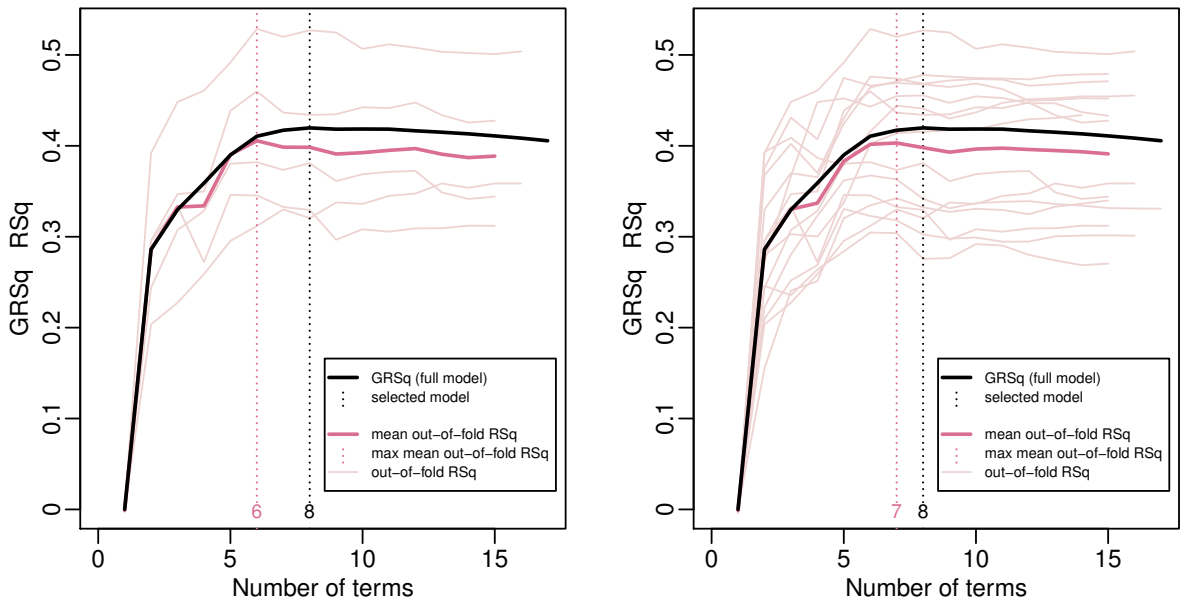

Figure 16:
Left *Earth model built with* `nfold=5`.
The pink lines show the out-of-fold R^2 for each fold. The thick red line is the mean of the pink lines. The black line is the usual GRSq line for the full model, ignoring cross-validation.
Right *Same but with* `ncross=3, nfold=5`. *There are 15 folds in all.*

10.1 What is the best value for `nfold`?

The question of choosing the number of cross-validation folds remains an open research question. We can only suggest that you try 5- or 10-fold cross-validation, unless you have a small dataset. With a small dataset some experimentation may be needed to get plausible results. Some gnarly details are discussed in the next chapter (for example Section 11.4 "Bias of cross-validation estimates").

10.2 The `ncross` argument

If we run `earth` twice with the same `nfold` argument we will get different cross-validation results, because earth randomly splits the data into folds differently each time. To average out this variation for more stable results, use the `ncross` argument to repeat the whole process of taking `nfold` folds multiple times. Example (right side of Figure 16):

```
fit <- earth(survived ~ ., data=etitanic, degree=2,
             keepxy=TRUE, ncross=3, nfold=5)
plot(fit, which=1, col.rsq=0)
```

10.3 Plotting cross-validation results

If you want `plot.earth` to show cross-validation statistics, use `keepxy=TRUE` so earth calculates the `oof.rsq` for every model size in every fold. That's why we use `keepxy=TRUE` in nearly all the examples in this chapter. More example plots may be found in Section 9.5.

Furthermore, if you want to use `plot.earth` or `plotmo` on a fold model (a model in the earth model's `cv.list`), use `keepxy=2` in the call to earth. This saves the in-fold data with each fold model.

10.4 Tracing cross-validation

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

```
CV fold 3: CVRsq 0.622  n.oof 86 12%  n.fold.nz 384 41%  n.oof.nz 43 39%
```

shows that in cross-validation fold 3, the `CVRsq` for the fold model was 0.622, measured on the 86 observations in the out-of-fold set.

The print also shows the number and percentage of non-zero values in the observed response in the in-fold and out-of-fold sets. This is useful if we have a binary or factor response and want to check that we 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 occurrences of any given level in the response constant across folds.

10.5 Two ways of collecting R^2

Earth uses two ways of collecting the R^2s generated during cross-validation (it can be confusing). The names used for these are `CVRsq` and `oof.rsq`. The names are somewhat arbitrary, but used consistently in earth's code and documentation. See Figure 17 for an overview.

(i) A folds's `CVRsq` is calculated from predictions made on the out-of-fold observations using the model built from the in-fold data. Earth builds the fold model like any other earth model, selecting the optimum number of terms using the GCV of the training (in-fold) data. The mean of these per-fold `CVRsq`'s is the `CVRsq` printed by `summary.earth`.

(ii) The `oof.rsq`'s are primarily for model selection, i.e., for selecting the best number of terms. They aren't printed by `summary.earth`, but by default are plotted by `plot.earth` (the pink lines in Figure 16). The `CVRSq`s described in (i) above are a subset of these.

A `oof.rsq` is calculated for every model size in each fold. (The model size is the number of terms in the model.) For each fold, it is calculated from predictions made on the out-of-fold observations using a model built with the in-fold data, after the model is pruned to the given size.

The `oof.rsq`'s are calculated only when `keepxy=TRUE` or `pmethod="cv"`. This is because calculating them is slow — earth has to call `update.earth` and `predict` for every model size in every fold.

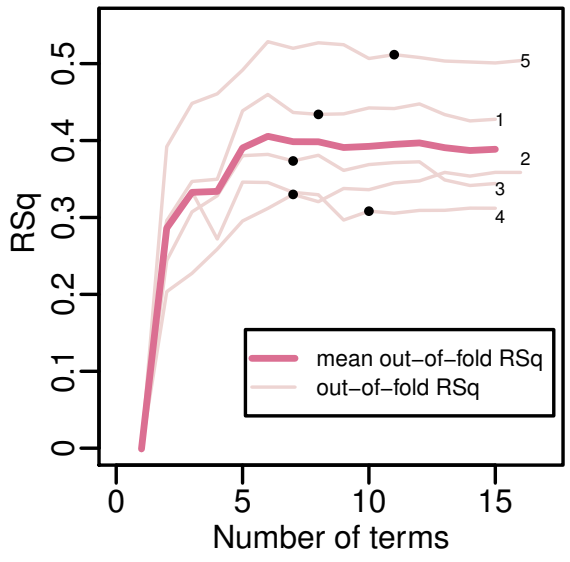

Figure 17: *Five-fold cross-validation of an* `earth` *model.*

This is the same model as the left side of Figure 16, but here we display slightly different information.

A black dot show the `CVRsq` *at each fold. This is the out-of-fold R^2 for the selected model at the fold (where the model is selected using the* `GRSq` *on the in-fold training data.) The* `CVRsq` *printed by* `summary.earth` *is the mean vertical position of these dots.*

The thick red line is the mean of the pink lines (the mean `oof.rsq`*).*

10.6 Using cross-validation to select the number of terms

Specify `pmethod="cv"` to select the number of MARS terms using cross-validation. Example (see Figure 18):

```
fit <- earth(survived ~ ., data=etitanic, degree=2,
             pmethod="cv", ncross=3, nfold=5)

plot(fit, which=1, col.rsq=0)
```

In Figure 18, we see that the GCV would have chosen 8 terms, but cross-validation chose 7 terms. (But since the curves are quite flat around that number of terms, cross-validation may choose a different number of terms if a different random seed is used before running earth. In practice, a higher value of `ncross` is necessary to get stable results with `pmethod="cv"`.)

Caveat: With `pmethod="cv"`, we are using the same data to *tune* and *evaluate* the model (in the printed summary statistics). Therefore the CV statistics printed by `summary.earth` may be optimistic. See Section 11.2 "Confusing parameter-selection fitness with final test fitness".

This is what happens internally when we specify `pmethod="cv"`. During cross-validation, earth measures the out-of-fold R^2 for each fold model (the pink lines in the figure below). The mean out-of-fold R^2 is then calculated for each model size (the red line). The optimum number of terms is taken to be at the maximum mean out-of-fold R^2 (the highest point of the red line). Earth then rebuilds the model using all the data, choosing the model in the backward pass at this optimum number of terms.

Notice that in the figure the pink lines for some fold models are truncated at the right. This is because the maximum number of terms for those models happens to be less than the 16 terms of the full model.

The red mean line is also truncated. For a given number of terms, half or more of

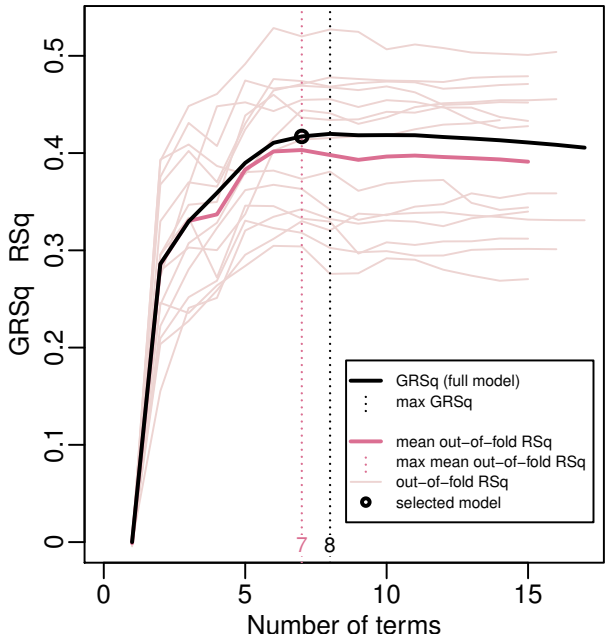

Figure 18: *Selecting a model with* `pmethod="cv"`.

This is identical to the right side of Figure 16, except that here there is a black circle at the selected model, and the legend is slightly different.

In practice, for stable model selection a higher value of `ncross` *is recommended than the value* `3` *used here for illustration.*

the pink lines have to be present for the mean to be considered valid. This prevents instability in the mean at the right of the plot.

10.7 Cross-validation statistics returned by earth

Section 10.5 described the R^2s collected during cross-validation. This section describes various additional cross-validation statistics.

Each of these is measured on the out-of-fold set for each fold, and summarized by averaging across all folds (except that `MaxErr` is "summarized" by taking the worst error across all folds). Use `summary.earth` to see the summary statistics and their standard deviation across folds.[1] See the `Value` section of the earth help page for more details of these statistics.

The statistics are:

- `CVRsq` See Section 10.5 Part (i).
- `oof.rsq` See Section 10.5 Part (ii).
- `MaxErr` Signed max absolute difference between the predicted and observed response. This is the maximum of the absolute differences, multiplied by `-1` if the sign of the difference is negative. The "summary" `MaxErr` is the worst `MaxErr` across folds.
- `ClassRate` (discrete responses only) Fraction of out-of-fold observations correctly classified. For binary responses a decision threshold of `0.5` is used.

If we cross-validate a binomial or poisson model (specified using earth's `glm` argument), earth returns the following additional statistics:

- `MeanDev` Deviation divided by the number of observations.
- `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.

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. 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 GLM statistics can be found in the following (and other) references: Pearce and Ferrier [21], Fawcett [6], and Harrell [10]. See the source code in `earth.cv.lib.R` for details of how the statistics are calculated, based on code kindly made available by Jane Elith and John Leathwick.

[1] We emphasize that `summary.earth` prints the deviation *across folds*, not the unknown deviation across hypothetical samples. See Section 11.5 "Variance of cross-validation estimates". It is easy to confuse the two.

10.8 An example: training versus generalization error

Figure 19 is reproduced from Figure 7.1 in Hastie et al. [12]. The figure was generated from models built with 100 training sets generated synthetically.

Figure 20 is an example illustrating some of the same principles. Instead of using new data, we use cross-validation. (Also, we use earth and the `mtcars` data.) The figure was generated with the following code:

```
fit <- earth(mpg~., data=mtcars, ncross=10, nfold=2, keepxy=TRUE)

plot(fit, which=1,
     col.mean.infold.rsq="blue", col.infold.rsq="lightblue",
     col.grsq=0, col.rsq=0, col.vline=0, col.oof.vline=0)
```

Figure 20 is "upside down" with respect to Figure 19 because it plots model perfor-

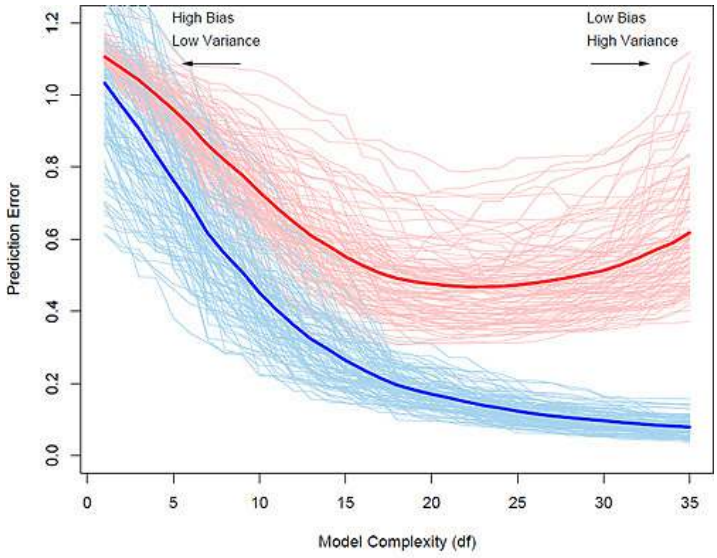

Figure 19: *Reproduced from Figure 7.1 of Hastie et al. [12]*

For models built from a 100 training sets, the pale blue lines show the prediction error measured on the training set. The pink lines show the error measured on a very large independent test set.

The thick lines average the pale lines to show the expected error.

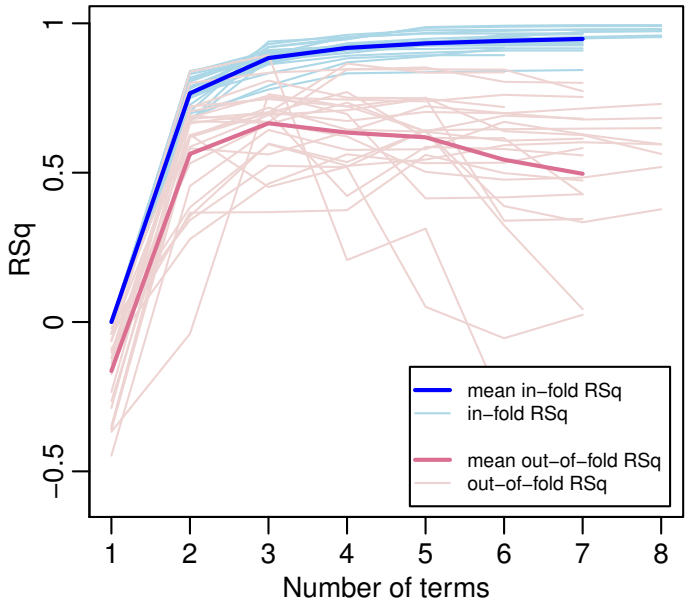

Figure 20: *Cross-validation of an earth model on the* `mtcars` *data.*

The Model Complexity along the horizontal axis in the figure above becomes the Number of Terms in this figure.

mance, not lack-of-performance. But we still see the same basic structure: the performance measured on the in-fold *training data* increases as we increase model complexity; on *independent data* the performance peaks and then decreases.

The maximum mean out-of-fold R^2 is at 3 terms, which in this example coincides with the number of terms selected by the GCV of the full model (set `col.grsq` to see this, not shown here).

10.8.1 Remarks

We mention first that several of the arguments in the call to `plot.earth` above simply remove display elements. The defaults for these arguments are inappropriate for this somewhat unusual plot (we aren't usually interested in the in-fold R^2s).

Note how variation of the pink lines increases with the number of terms. The more flexible the model, the more it overfits to randomness in the training set, and thus more randomness enters its estimation of R^2 on independent data.

Much of the variation of the pink curves in Figure 20 is due to the relatively small size of the out-of-fold datasets. If we measured R^2 on a very large test set (instead of the out-of-fold data) we would still see variation, but much less.

For each pink curve, we would see more variation if we used genuinely independent training sets (rather than the pseudo-independent cross-validation fold data).

The `mtcars` dataset is small (32 observations). Only two folds were used above (but repeated 10 times with `ncross`) to keep the out-of-fold sets large enough for somewhat stable results. With this small `nfold`, cross-validation bias may be an issue, because of the small size of the in-fold sets relative to the full dataset. So the R^2 on the out-of-fold data will tend to be smaller than it would be across full-sized independent samples. See Section 11.4 "Bias of cross-validation estimates".

For a nice illustrative graph we used `degree=1`, but for the `mtcars` data a higher `degree` is actually more appropriate.

11 Understanding cross-validation

This chapter tries to clarify some aspects of cross-validation. It was written in response to email about cross-validating earth models. The chapter is mostly a general discussion, not limited to earth models. The exposition takes a frequentist approach, using arguments based on hypothetical situations where we have access to extra data.

For a description of cross-validation, see for example Hastie et al. [12], Section 7.10, Duda et al. [4] Section 9.6, or even Wikipedia
`http://en.wikipedia.org/wiki/Cross-validation_(statistics)`.
An in-depth reference is Arlot and Celisse [1].

11.1 Datasets for measuring performance

In the next section we will discuss some common cross-validation mistakes. But first we review some aspects of measuring a model's performance, and the datasets needed to do that. Understanding the role of these datasets is important for applying cross-validation correctly.

Typically we want to measure our model's *generalization* performance, and so want to measure prediction error on new data, i.e., not on the training data. (If that isn't immediately obvious, see FAQ 13.7.) We typically want to measure performance in two scenarios:

(a) For **parameter selection**, i.e., to choose certain key model parameters during the model building process. For example, for earth models we need to select the best number of terms (so the parameter here is the number of terms). Another example is `rpart` trees, where we need to choose the optimum tree size. The new data are used as *parameter-selection data*, also commonly called *model-selection* or *validation* data.

(b) For **model assessment**, i.e., to measure the performance of the final model. Here the new data are used as *test data*.

We thus require three independent datasets:

(i) the **training** data,
(ii) the **parameter-selection** data for (a) above,
(iii) the **test** data for (b) above.

The ideal way to meet these requirements is to actually have large amounts of new data drawn from the same population. Usually we don't actually have access to such data, and so must resort to other techniques.

One such technique is the GCV used when building MARS models, which bypasses the need for model-selection data by using a formula to approximate the RSS that would be measured on new data.

Another technique, more universal, is cross-validation. Depending on how it is used, cross-validation can emulate either the parameter-selection or test data.

A note on linear models. Many of us started learning about statistical modeling by studying *linear models*. It's reasonable to ask why we don't bother with all the above datasets with linear models. The answer is that linear models are relatively inflexible, and with these simple models the difference between the performance measured on the training data and on independent test data is inconsequential, provided certain assumptions are met. So we don't need a separate test set. Additionally, when building a linear model there is no separate parameter-selection step, so we don't need a parameter-selection set. This will no longer be true if we adopt a more flexible modeling procedure. For example, if we are doing variable selection for a linear model (or if we have a large number of possibly irrelevant variables), we should measure R^2 on data that is independent of the training data, or adjust R^2 as if we had such data (by using measures such as the AIC). With more flexible models overfitting becomes more likely, since a flexible model can adapt to the peculiarities of the training data.

11.2 Common cross-validation mistakes

In this section we list some mistakes that are easy to make when cross-validating. All of these make the model's performance seem better than it is. Given how easy it is to make these mistakes, a certain amount of skepticism is warranted when papers present final model assessment statistics based on cross-validation.

Independence of observations

The out-of-fold data must play the role of new data. It's thus important that it out-of-fold data isn't "contaminated" by the in-fold training data. This implies that the observations must be independent.

Lack of independence means that the in-fold data used for training are partially duplicated in some sense in the out-of-fold data used for testing, and cross-validation will tend to give optimistic fits. At a minimum, we should avoid "twinned" observations.

Pre-tuning

Don't use a set of data for selecting model parameters, then use the same data for subsequent cross-validation of the model. Cross-validation must be applied to the entire model building process. Any parameter that is tuned to the training data must not be tuned before cross-validation begins.

A word of explanation on the above paragraph. Let's say we do in fact optimize a parameter to the full dataset before cross-validation begins. By doing so, we are also to some extent optimizing to the out-of-fold data used during cross-validation (because the out-of-fold data are, after all, drawn from the full data). The out-of-fold data are thus contaminated and can no longer legitimately play the role of independent test data, and performance measured on the out-of-fold data will appear better than it should.

There are however contexts where it's acceptable to make decisions before cross validation. Decisions that are made independently of the training data are acceptable. Decisions about which variables to include that are made independently of the response are acceptable. See the comments near the end of Section 7.10.2 in Hastie et al. [12].

Confusing parameter-selection fitness with final test fitness

If fits obtained by cross-validation are used to select a model's parameters, then the final test fit quoted for the selected model must be recalculated for that model using a *new* set of independent data. The cross-validation fit used when selecting the model cannot be quoted as the fitness for that model—that would be optimizing for the test set by conflating the parameter-selection and test data.

11.3 What does cross-validation measure?

The mechanics of cross-validation are easy to understand. Understanding what cross-validation actually estimates involves some subtleties, and the rest of this chapter is somewhat technical.

Cross-validation estimates "average" not "conditional" error

Cross-validation doesn't really estimate the generalization performance of our model, built on a single set of data (which is usually what we want to know when applying a modeling technique like earth). Instead, cross-validation approximates the performance of our model building algorithm on a range of training sets. It approximates the average performance we would see in a hypothetical scenario where we build many models, each on a fresh sample of training data of approximately the same size as the original sample, and measure the performance of each of those models on independent test data (all data being i.i.d. from the same population).

In other words, cross-validation is better at estimating the expected prediction error across training sets, not the prediction error conditional on the training data we have at hand. In Figure 19 on page 45, our model is one pink line but cross-validation approximates the solid red line. See also Hastie et al. [12] Section 7.12 "Conditional or Expected Test Error?"

Some details. Cross-validation differs from the hypothetical scenario above because in cross-validation the training (in-fold) sets share observations and aren't as varied as they would be in the hypothetical scenario. Also, the training set of a fold incorporates test sets from other folds. This induces a relationship between the residual errors (or whatever is used to measure performance) across folds, particularly in the presence of outliers.

Expected value of R^2 across models

Consider the hypothetical scenario above, and let us measure performance as R^2 on an independent test set: for each model we draw a training set from the population, and also draw a test set from the population to measure R^2. If we built many models and took the average R^2 over all the models, we would eventually close in on a stable average R^2 value.

This average R^2 would be the same regardless of the size of the test sets, assuming we repeated the experiment enough times. In other words, the *expected value* of R^2 across models doesn't depend on the size of the test sets — but the *variance* of the R^2s certainly does, which leads to the next section.

Variance of R^2 across models

Once we have an estimate of the generalization performance of the model (such as R^2 on independent data), we typically want to know the *stability* of that estimate, usually expressed as the variance of the estimate.

Typically we want to know how our estimated R^2 would be expected to change if we had a different training set — the sampling variance of R^2. This is the variance we would measure across models in the hypothetical scenario above if the test sets were extremely large (so all variation is due to the training sets, not the test sets).

On the other hand, if the test sets are small, the variance of R^2 will include extra variation due to the small size of the test sets. This is the scenario emulated by cross-validation. So in cross-validation, it isn't possible in general to disentangle the variability due to the in-fold training sets and the variability due to the small out-of-fold test sets.

11.4 Bias of cross-validation estimates

Cross-validation tries to establish the quality of a model built on the full sample by using models built on *smaller* subsets of the sample (often 4/5 or 9/10 of the sample). Generally a model built with a subset will be "worse" than a model built with the full sample (a smaller training set is less desirable). Thus the cross-validation R^2 will tend to be lower than it should be[1]. That is, the cross-validation R^2 is conservatively *biased*.

[1]The *Cross-validation* R^2 here is the mean of the R^2 of each fold, each measured on the out-of-fold data. It is the `CVRsq` printed by `summary.earth`.

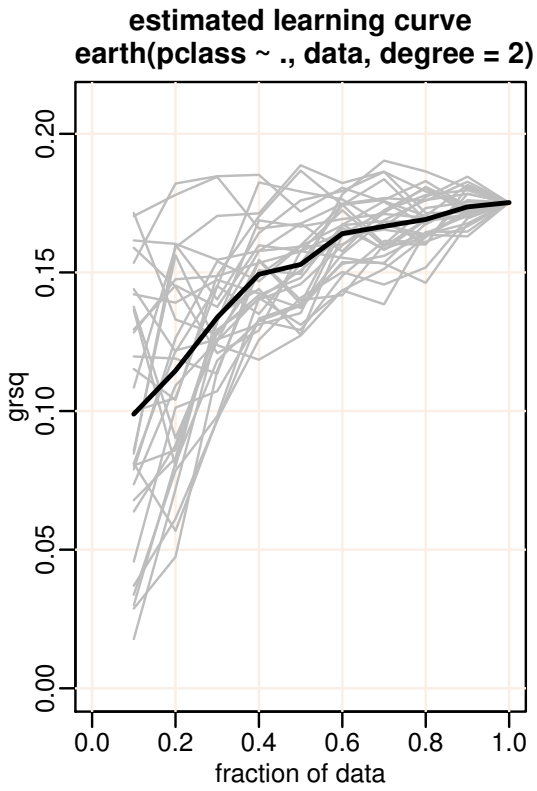

Figure 21: *The estimated learning curve of an earth model.*

The dark line is the mean of the gray lines.

Each gray line shows the GRSq for one set of subsets of the training data.

To see where the model sits on the learning curve (Figure 7.8 in Hastie et al. [12]), one technique for earth models is to plot the GRSq of models built with different sized subsets of the sample, and average out variation by repeating several times. Figure 21 gives an example, produced by the following R code. (You can ignore the code and just look at the figure.)

```
learning.curve <- function(data, func, field="grsq", ncurves=30)
{
    # set up the plot (call func on full data to establish ylim)
    body <- body(func) # needed only for the plot title
    plot(0, xlim=c(0,1), ylim=c(0, 1.2 * func(data)[[field]]), type="n",
         xlab="fraction of data", ylab=field, cex.main=1.1, xpd=NA,
         main=paste("estimated learning curve\n",
           substr(paste(deparse(substitute(body)), collapse=""), 1, 40)))
    grid(col="linen", lty=1)
    all.results <- rep(0, 10)
    for(curve in 1:ncurves) {                       # for each gray line
        sample <- sample.int(nrow(data))
        results <- double(10)
        for(fold in 1:10) {
            sub.data <- data[sample[1:(fold * nrow(data) / 10)],]
            results[fold] <- func(sub.data)[[field]]
        }
        lines((1:10)/10, results, col="gray")      # one gray line
        all.results <- all.results + results
    }
    lines((1:10)/10, all.results / ncurves, lwd=2) # the mean line
}

learning.curve(etitanic,
               function(data) earth(pclass~., data, degree=2))
```

From the figure, the bias is small enough with 10 folds (the curve is flat enough with 90% of the data).

Remember that this is just an *estimated* learning curve because it is created from a single training sample. Ideally we would like to estimate the learning curve using many fresh training sets. If we did that, we would see variation in the curves on the far right of graph which we don't see in Figure 21. So even with `nfold=5` the bias is acceptably small, relative to the (unknown) variance.

11.5 Variance of cross-validation estimates

Cross-validation bias seems to be much discussed, but usually a more serious problem with cross-validation is the *variance* of cross-validation estimates across samples, and our inability to estimate this variance. How much would we expect the cross-validation R^2 (averaged across folds) to change if we used a different training sample? (The new sample would be of the same size and drawn from the same population as the original.) It isn't much comfort to know that the *expected* value of an estimate is correct up to small bias, if the single estimate we have at hand could be far from that expected value because of estimation variance.

Quantifying the variance of cross-validation statistics in general is an ongoing research problem (see Bengio and Grandvalet [2] for an explanation of some of the difficulties involved). Unfortunately it isn't really possible to estimate the variance of the cross-validation R^2 of earth models.

An indication is given by the variance of the CV R^2 across folds (printed by `summary.earth` as a standard deviation). However, this variance includes extra variability because we are looking at the R^2 per fold instead of the mean R^2 across folds. On the other hand, it incorporates less variability due to training sets than if we actually used fresh training data at each fold. Also it is unstable because of the small size of the out-of-fold test sets (the variance of the variance is high).

For earth models, another indication is the variance of GRSq in the estimated learning curve (Figure 21). Assuming GRSq is an acceptable surrogate for R^2 on independent data, the variance across the gray curves gives an approximate lower bound of R^2 variance across models for variously sized training sets. We say "lower bound" because the estimated learning curve is created from only a single training set (if we used fresh data for each model the variance at the right of the curve wouldn't taper to zero).

12 Estimating variable importance

This chapter discusses how to estimate the relative importance of variables in an earth model.

12.1 Introduction to variable importance

What exactly is variable importance? A working definition is that a variable's importance is a measure of the strength of the relationship between observed values of the variable and the observed response. It is this measure of importance that the `evimp` function tries to estimate.

You might say that we can measure a variable's importance by changing the variable's value and measuring how the response changes. Indeed, the fact that an earth model is represented by an equation seems to imply that this is the way to go. However, except in special situations, there are problems with this way of thinking because:

(i) It assumes we can change the variable, which usually isn't 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 can occur in isolation. In practice, a variable is usually tied to other variables, and a change to the variable would never occur in the population without simultaneous changes to other variables. For example, in the `trees` data, a change in a tree's height is associated with a change in the tree's girth.

(iii) It implies a causal relationship, which often isn't the case. Changing the amount of mud doesn't tell us anything about the amount of rain.

Thus it's better to think in terms of the effect of the variable on the response averaged over the entire population. That is to say, the *expected* effect. In practice, we have to figure out how to use the model and the sample as a surrogate for the population, which isn't trivial.

Note that variable importance in the *equation that MARS derives from the data* isn't really what we have in mind here. 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 doesn't even appear in the equation). Section 12.5 has a few words on how to use `plotmo` to estimate variable importance in the MARS equation.

12.2 Estimating variable importance

Estimating predictor importance is in general a tricky and even controversial problem. There is usually no completely reliable way to estimate the importance of the variables in a standard MARS model. The `evimp` function just makes an educated (and in practice useful) estimate as described below.

12.3 Three criteria for estimating variable importance

The `evimp` functions uses three criteria for estimating variable importance in a MARS model.

(i) The `nsubsets` criterion counts the number of model subsets that include the variable. Variables that are included in more subsets are considered more important. This is the criterion used by `summary.earth` to print variable importance.

By "subsets" we mean the subsets of terms generated by earth's backward 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.

(ii) The `rss` criterion first calculates the decrease in the RSS for each subset relative to the previous subset during earth's backward pass. (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, for ease of interpretation the summed decreases are scaled so the largest summed decrease is 100. Variables which cause larger net decreases in the RSS are considered more important.

(iii) The `gcv` criterion is the same, but uses the GCV instead of the RSS. Note that adding a variable can sometimes *increase* the GCV. (Adding the variable has a deleterious effect on the model, as measured in terms of its estimated predictive power on unseen data.) If that happens often enough, the variable can 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, because `evimp` calculates *relative* importances.

12.4 Example

This code

```
earth.model <- earth(O3 ~ ., data=ozone1, degree=2)
evimp(earth.model, trim=FALSE)  # trim=FALSE to show unused variables
```

prints the following (your version of earth may give slightly different results):

```
           nsubsets   gcv    rss
temp             11 100.0  100.0
humidity          9  35.7   38.9
ibt               9  35.7   38.9
doy               8  33.7   36.5
dpg               6  26.2   28.7
ibh               5  31.1>  33.0>
vis               5  21.2   23.9
wind              2   9.3   11.9
vh-unused         0   0.0    0.0
```

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. (Unused variable are printed here because we passed `trim=FALSE` to `evimp`. Normally they are omitted from the print.)

The `nsubsets` column is the number of subsets that included the corresponding variable. For example, `temp` appears in 11 subsets and `humidity` in 9.

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

A "`>`" is printed after `gcv` and `rss` entries that increase 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.

12.5 Estimating variable importance in the MARS equation

Running `plotmo` 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 (Section 12.1).

12.6 Using `drop1` to estimate variable importance

As an alternative to `evimp`, we can use the `drop1` function in the standard `stats` package. (To do this, we must build the earth model using the formula interface, not the `x,y` interface.) Calling `drop1(earth.model)` will delete each predictor in turn from the model, rebuild the model from scratch each time, and calculate the GCV each time. We will get warnings that the earth library function `extractAIC.earth` is returning GCVs instead of AICs — but that is what we want so we can ignore the warnings. (Turn off just those 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 isn't much use in this context.

Remember that this technique only tells us how important a variable is with the other variables already in the model. It doesn't tell us the effect of a variable in isolation.

We will get lots of output from `drop1` if we built the original earth model with `trace>0`. We can set `trace=0` by updating the model before calling `drop1`. Do it like this: `earth.model <- update(earth.model, trace=0)`.

12.7 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 partially averaged out by building a bagged earth model and take 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 Max Kuhn's `caret` package [14].

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

12.8 Remarks on `evimp`

The `evimp` function is useful in practice but the following issues can make it misleading.

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 earth model building algorithm 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.

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

For factor predictors, importances are estimated on a per-level basis (because earth splits factors into indicator columns, essentially treating each level as a separate variable). The `evimp` function should have an option to aggregate the importances over all levels, but that hasn't yet been implemented.

TODO Enhance `evimp` to use cross-validation statistics when available.

13 FAQ

13.1 What are your plans for earth?

Support of NAs would be good.

Multicore support would make earth significantly faster.

The `weights` argument makes earth very slow with big models.

The variance model code could be factored out of earth into its own package, allowing prediction intervals for "any" model.

13.2 How do I cite the earth package?

Cite the `earth` package like this:

> S. Milborrow. Derived from mda:mars by T. Hastie and R. Tibshirani.
> *earth: Multivariate Adaptive Regression Splines*, 2011. R package.

The following BibTex entry does the trick. The extra curly braces in the author field are necessary to get BibTex to order the entry correctly on the last name of the first author. The `url` field is optional.

```
@Manual{earthpackage,
  title = {earth: Multivariate Adaptive Regression Splines},
  author = {S. {Milborrow. Derived from mda:mars by T. Hastie and R. Tibshirani.}},
  year = {2018},
  note = {R package},
  url = {\url{https://CRAN.R-project.org/package=earth }}
}
```

From within R you can use the following (but you will have to massage the results to get BibTex to order the entry correctly, because of the unusual author field):

```
> library(earth)
> citation("earth")
```

13.3 What are the limits on earth model size?

As an example, the earth test suite has additive models with 8 million cases and 100 variables, and 80 million cases and 2 variables. Bigger models are possible. See Section 2.7.

13.4 Can I use earth with a binary response?

Yes. Typically you want to predict a probability. Usually the best way to proceed is:

(i) Convert your binary response to a `logical`, if it isn't that already. To do the conversion, you can use code like this:

```
data$response <- data$response == "survived"
```

Actually, you don't necessarily have to do that. Although a response of type `logical` is canonical in this situation, if you prefer you could also use a two-level factor or a 1s-and-0s response. Earth doesn't mind.

(ii) Include `glm=list(family=binomial))` in your call to `earth`. This tells earth to build a model that estimates response probabilities.

See Section 4.1(i) for an example, and the rest of that chapter for details.

13.5 Can I use earth with a categorical response?

Yes. If your response has only two categories (it's binary) please see the above FAQ.

If your response has more than two levels, make sure your response is an unordered R `factor`. For an example, see Section 4.1(ii), and the rest of that chapter for details.

Before handing it to its internal engine, earth will expand your response to multiple columns (an indicator column for each factor level) and build a multiple-response model. This is described in Chapter 5.

13.6 What is a GCV, in simple terms?

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

Usually 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 introduces other complications and 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 isn't that good, it is usually good enough for comparing models during the backward pass.

The GRSq measure used in the earth package standardizes the raw GCV, in the same way that R^2 standardizes the RSS (FAQ 13.11).

GCVs were introduced by Craven and Wahba [3], and extended by Friedman and Silverman [7,9]. See Hastie et al. [12], Section 7.10 "Cross-Validation", and the Friedman MARS paper [7]. GCV stands for "Generalized Cross Validation", a perhaps misleading term because no cross-validation is actually performed.

13.7 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 won't give good predictions on new data. The idea is that the training data capture 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 is thus too specific to that 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 (viz. the intercept and regression coefficients) 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, and we don't need GCVs.

For linear models this is no longer true if we do automatic variable selection (or if we have a large number of possibly irrelevant variables) — because the process of selecting variables increases the flexibility of the model. This is why measures like the AIC are necessary — as used in, say, `stats::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.

13.8 How do I get p values for earth model coefficients?

You can't get meaningful p values for MARS models. If the `GRSq` is satisfactory, we can have reasonable confidence in the model *as a whole*, but we cannot assign confidence levels to parts of the model in *isolation*. Perhaps we may suppose we can get p values like this:

```
earth.mod <- earth(Volume ~ ., data = trees) # standard earth model
bx <- earth.mod$bx                    # earth's basis mat
lm.mod <- lm(trees$Volume ~ bx[,-1]) # standard linear regression on earth's basis mat
                                      # (-1 drops intercept)
summary(lm.mod)                       # prints p values -- meaningless in this context
```

That will indeed print p values, but they are meaningless. The p values (actually t values) and standard errors will be small, indicating that all the predictors are important. This is a self-fulfilling prophecy: if the terms weren't important, the MARS algorithm wouldn't have included them in the model.

The summary formulas used internally by `summary.lm` assume that the terms are predefined. They ignore the uncertainty in the selection of the terms, and don't discount that uncertainty when printing the p values.

Put another way, the null hypothesis in `summary.lm` for each term is essentially that the term is unimportant — but the MARS algorithm selects only meaningful terms, so the p values returned by `summary.lm` will always be small.

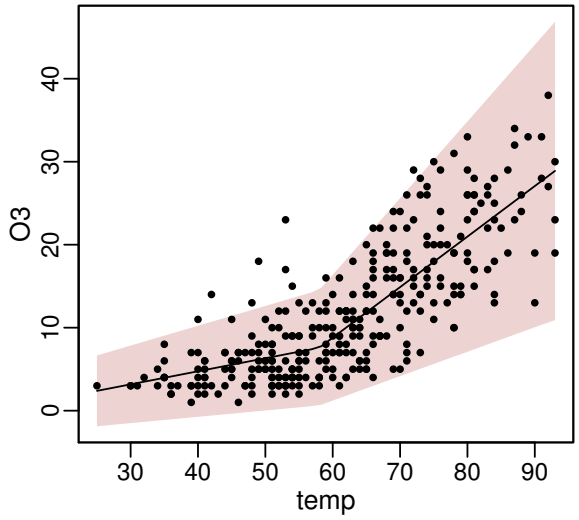

Figure 22: *Prediction intervals on an earth model, produced with the following code:*

```
mod <- earth(O3~temp, data=ozone1,
             nfold=10, ncross=30,
             varmod.method="lm")

plotmo(mod, pt.col=1, level=.95)
```

13.9 Can I get confidence intervals on my predictions?

Yes, prediction intervals are available for earth models. Use the `varmod.method` argument (Figure 22).

Getting plausible intervals may require a bit of work. Please see the vignette "Variance models in earth". I suggest you start off with `varmod.method="lm"` and see if that gives you plausible results — plot the model and check the prediction bands in the residual plot.

Note however that confidence levels on the earth *coefficients* are not available (FAQ 13.8).

13.10 Can R^2 be negative?

Yes, R^2 (`rsq`) can sometimes be negative when both the following are true:

(i) the test set is not the training set (for example, in cross-validation), and,

(ii) we use the general definition of R^2

```
rsq = 1 - rss / tss,
```

where `rss = sum((y - yhat)^2)` is the residual sum-of-squares and `tss = sum((y - mean(y))^2)` is the total sum-of-squares. This is the definition used in the earth code (there are a few other definitions, further discussion later in this section).

The simplest example where we see an negative R^2 is an intercept-only model. (An intercept-only model always predicts a constant value, the mean of the training data response.) An intercept-only model will give a negative R^2 on test data, unless the training data and test data happen to have the same mean. Why is this? On the test data, the intercept-only model predicts, as always, the mean of the *training* data. Thus on the test data the residuals will be greater on the whole than if we predicted the mean of the test data.[1] That is another way of saying that on the test data the

[1]Recall that $\sum_i (x_i - \mu)^2$ is minimized when μ is the mean of the x_i's.

residual sum-of-squares will be greater than the total sum-of-squares, and `rsq = 1 - rss / tss` will be negative. Examples can be seen in the left of Figure 20 on page 45 (the pink lines are below zero in that region of the plot).

There is actually more than one definition of R^2. You may be more familiar with the definition

```
rsq = regression.sum.of.squares / tss
```

which is indeed always non-negative. With R^2 measured on the training data, the definitions are equivalent for linear regression (with an intercept) and for earth models. But the definition of R^2 used by earth generalizes more easily for measuring performance on data not in the training set, and for measures like the `GRSq`.

The "squared" in "R-Squared" is misleading in that it implies a non-negative value. Perhaps we should use the alternative term "coefficient of determination", but "R-Squared" is common.

13.11 Can GRSq be negative?

Yes, can GRSq go negative.

The statistic `GRSq` is earth's estimate of the generalization performance of the model. It is defined, analogously to R^2 (FAQ 13.10), as

```
GRSq = 1 - GCV / GCV.null,
```

where `GCV.null` is the GCV of an intercept-only model.

A negative `GRSq` indicates a severely over-parameterized model — a model that wouldn't generalize well, even though it may be a good fit to the training data. During earth model building, `GRSq` can become negative. However, after the backward pass the model will end up with a non-negative `GRSq`.

Adding a term to the model will always increase the R^2 on the training data (up to the limits of numerical accuracy). But adding that term could reduce the estimated

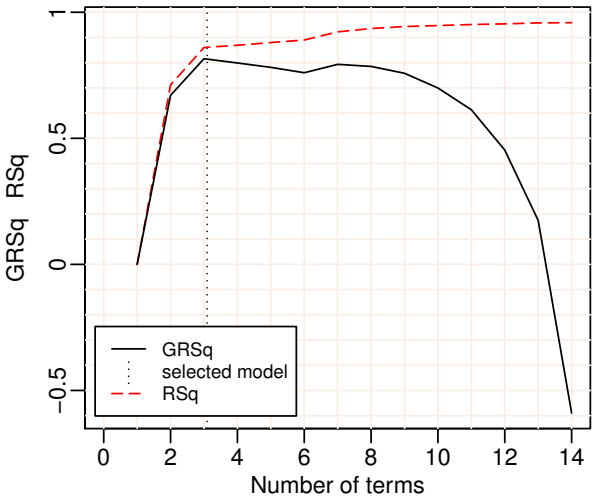

Figure 23: *Negative GRSq's.*

After term 3, adding a term reduces the estimated predictive power of the model, apart from term 7. By term 14 the GRSq is negative.

The R^2 on the training data always increases as earth adds terms (dashed line).

predictive power of the model on new data, and would thus *decrease* the GCV, and thus also GRSq. (We see that happening for later terms in almost any earth model Selection graph.) Decrease GRSq often enough and it will eventually become negative. Watch the GRSq take a nose dive in this example (Figure 23):

```
fit <- earth(mpg~., data=mtcars, trace=4)
plot(fit, which=1, col.npreds=0, xlim=c(0,14),
     col.sel.grid="linen", legend.pos="bottomleft")
```

In severe cases, GRSq might even be set to -Inf, which brings us to the following FAQ.

13.12 Why does "Termination condition: GRSq -Inf" mean?

It's not something to especially worry about. It means that the forward pass stopped adding terms because the GRSq got so bad that it was pointless to continue. It's similar to Termination condition: GRSq -10 (Section 3 iv) except that the GRSq was even worse than -10. After the backward pass, GRSq will be non-negative.

Some details. Earth sets the GCV to Inf during model building if the effective number of parameters is greater than the number of observations. Under these conditions, the GCV no longer approximates the leave-one-out RSS. To see this, consider the formula for the GCV

```
GCV = RSS / (n * (1 - nparams / n)^2))
```

where n is the number of observations. From the formula we see that the GCV increases and then decreases if nparams / n approaches and then exceeds 1 as terms are added to the model. To prevent this undesirable non-monotonic behavior, if nparams / n >= 1 then earth doesn't use the formula but instead directly sets the GCV to Inf.

13.13 How is the default number of terms nk calculated?

The default nk is somewhat arbitrary. Thus if print.earth shows that the termination condition for our model is Reached maximum number of terms, we might consider increasing nk. Perhaps the forward pass was terminated too soon.

In detail, the default is:

```
nk = min(200, max(20, 2 * ncol(x))) + 1
```

In words: this doubles the number of predictors, forces that into the range of 20 to 200, and finally adds 1 for the intercept.

The numbers 20 and 200 are fairly arbitrary. The lower limit of 20 seems reasonable for situations where we have just a few predictors. The upper limit of 200 helps prevent excessive memory use in the forward pass. Typically we will reach one of the termination conditions long before we reach 200 terms. (The termination conditions are described in Chapter 3 "Termination conditions for the forward pass".)

13.14 Why do I get fewer terms than nk, even with pmethod="none"?

There are several conditions that can terminate the forward pass, and reaching `nk` is just one of them. See Chapter 3 "Termination conditions for the forward pass". The various stopping conditions mean that the actual number of terms generated by the forward pass will usually be less than a big `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. To see this behavior, run earth with `trace=2` or higher. You will see in the `Terms` column that although earth usually adds two terms it will sometimes add just one.

(ii) As a final step (just before the backward pass), the forward pass deletes linearly dependent terms, if any, so all terms in `dirs` and `cuts` are independent. If this happens and tracing is enabled you will get a message like `Fixed rank deficient bx by removing 3 terms`.

And remember that the backward pass will usually discard further terms unless `pmethod="none"`.

13.15 Why do I get fewer terms than my specified nprune?

The backward 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 to get exactly `nprune` terms by specifying `penalty=-1`. An example:

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

This special value of `penalty` causes earth to set the GCV to `RSS/nrow(x)`. Since the RSS on the training set always decreases with more terms, the backward pass will choose the maximum allowable number of terms. (Unless one of the other termination conditions is met beforehand, Section 3.)

Setting `nprune` is really only for advanced use of earth—see the comment at the end of the following FAQ.

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

If you want a smaller model than that built by default, it's usually best to generate a big set of basis functions in the forward pass (by specifying a big `nk`) and prune these back (by specifying a small `nprune`). This is generally better than directly building a small model by specifying a small `nk` — the backward pass can choose any of the available terms to include, whereas the forward pass can only see one term ahead.

However, it must be asked: Do we really want to force a small model? Usually we want the best predictive model rather than a small model, unless we have special reasons otherwise. By forcing `nprune` we are affecting the ability of the MARS algorithm to produce a model with good generalization ability.

13.17 Which predictors are used 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)))
}
```

13.18 Which predictors were added to the model first?

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

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

which will list the terms remaining after the backward pass, in the order they were added by the forward pass.

It should be remarked that the order in which terms or predictors are added by the forward pass isn't necessarily indicative of their relative importance. Consider using the `evimp` function.

13.19 `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 Chapter 5 "Factors (categorical variables)".

13.20 How does `summary.earth` order terms?

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

With the default `decomp="anova"`, the terms are ordered as follows:

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

(v) and finally, within term pairs the term for "predictor less than hinge" is placed before the term for "predictor greater than hinge" (so for example, `h(16-Girth)` is before `h(Girth-16)`).

The reordering is done by the function `reorder.earth`.

13.21 How do 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 <- sort(sample(1:nrow(trees), .8 * nrow(trees)))
test.subset <- (1:nrow(trees))[-train.subset]

earth.model <- earth(Volume ~ ., data = trees[train.subset,])

# print R-Squared on the test data
print(summary(earth.model, newdata=trees[test.subset,]))

# manually calculate R-Squared on the test data (same as above call to summary)
yhat <- predict(earth.model, 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 (call `set.seed` to prevent this randomness).

Also, remember that the test set shouldn't be used for parameter tuning because you will be optimizing for the test set (Section 11.1). Instead use GCVs, separate parameter-selection sets, or techniques such as cross-validation with earth's `nfold` parameter. (Cross-validation repeats the process in the above code five or ten times, using a different subset each time).

13.22 Why is `plot.earth` not showing the cross-validation data?

Use `keepxy=TRUE` in the call to `earth` (as well as `nfold`). See Section 10.3.

13.23 How do I add a plot to an existing page with `plot.earth` or `plotmo`?

Use `do.par = FALSE`, otherwise these plotting functions start a new page.

13.24 What about bagging MARS?

The `caret` package [14] provides functions for bagging earth (and for parameter selection). Our personal experience has been that bagging MARS doesn't give models with better predictive ability (probably because the MARS algorithm is fairly stable in the presence of perturbations of the data, and bagging works best for "unstable" models). Your mileage may vary (we would be interested if it does). We tested just a couple of datasets, but did try a few different approaches, including using a modified version of earth that randomized the set of variables available at each forward step to increase variability (similar to random forests).

13.25 Why do I get `Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred`?

You will only see this warning when using earth's `glm` argument. You can safely ignore the warning in an earth context. The GLM coefficients for the model terms may be very large, but it doesn't matter — the predictive ability of the model is unimpaired.

The warning is issued when `glm.fit` generates a model that perfectly separates the classes in the training data. A perfect fit is usually considered a good thing, not something that should cause a warning. However, the warning is issued because certain model statistics (such as the t-values) generated by the mathematics inside `glm.fit` will be unreliable for subsequent inference on the model. That doesn't matter in an earth context, because earth doesn't use those statistics. (And, anyway, the t-values are meaningless even when the warning isn't issued, because of the amount of processing done by earth to generate the terms before it calls `glm.fit`, FAQ 13.8.)

The warning message is more likely to occur during cross-validation (using earth's `nfold` parameter). With cross-validation we are looking at more, and smaller, datasets, so the chance of a perfectly separable set is more likely. If the warning is issued, the coefficients for the terms of the fold model may be very large, but they aren't used in the final model anyway. The cross-validation statistics calculated by earth (such as `CVRsq`) remain valid.

Note added Dec 2014: Andrew Gelman gives an example where this message is given because of non-convergence of the glm algorithm. I believe this is rare in practice. `http://andrewgelman.com/2011/05/04/whassup_with_gl`.

13.26 Why do I get `Error: XHAUST returned error code -999`?

The short answer: you should never see the above message (fixed in earth version 2.6-0). If you do, please let us know.

One workaround is to change `pmethod` from `"exhaustive"` to `"backward"`.

You can also try the following. These instructions work on the assumption that the default value of `Exhaustive.tol` is too big for your dataset. First please read the description of the `Exhaustive.tol` argument in the `Arguments` section of the earth

help page. Then run earth with `trace=1`, so earth prints the reciprocal of the condition number of the earth basis matrix `bx`. (The condition number here is the ratio of largest to the smallest singular value of `bx`.) Then set `Exhaustive.tol` to greater than the printed value (something like `Exhaustive.tol=1e-8`), and run earth again. Now earth will automatically change `pmethod` from `"exhaustive"` to `"backward"` when necessary to avoid the above error message.

It must be said that it is hard to believe under these conditions that the resulting model will be much good. The data don't allow a decent predictive model to be built.

Some details. Certain data cause collinearity in the earth basis matrix `bx` which slips by the usual checks. This causes the `leaps` routine to fail. The usual checks are:

(i) while building the basis matrix, the C code does a check to drop collinear terms (`BX_TOL` and `QR_TOL` in the C code)

(ii) after building the basis matrix, the C code drops any remaining collinear terms (`RegressAndFix` in the C code)

(iii) the `leaps` Fortran routine `sing` checks for collinearity.

Some data get through all these tests, probably because we are near the numerical noise floor and numerical rounding is essentially changing the data randomly. When `pmethod="exhaustive"`, earth performs an SVD of `bx`, and as a last resort if the condition number is out-of-range forces `pmethod` from `"exhaustive"` to `"backward"`.

References

[1] S. Arlot and A. Celisse. *A Survey of Cross-Validation Procedures for Model Selection.* Statistic Surveys, 2010. Cited on page 47.

[2] Y. Bengio and Y. Grandvalet. *No Unbiased Estimator of the Variance of K-Fold Cross-Validation.* J. Mach. Learn. Res., 5:1089-1105, 2004. `http://jmlr.csail.mit.edu/papers/v5/grandvalet04a.html`. Cited on page 52.

[3] P. Craven and G. Wahba. *Smoothing Noisy Data with Spline Functions.* Numer. Math 31, 377-403, 1979. Cited on page 58.

[4] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification.* Wiley-Interscience Publication, 2000. `http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0471056693.html`. Cited on page 47.

[5] Julian Faraway. *Extending the Linear Model with R.* CRC, 2005. `http://www.maths.bath.ac.uk/~jjf23`. Cited on page 5.

[6] Tom Fawcett. *ROC Graphs: Notes and Practical Considerations for Researchers. Revised version of Technical report HP Laboratories.* HP Labs, 2004. `http://biostat.mc.vanderbilt.edu/twiki/bin/view/Main/RmS`. Cited on page 44.

[7] Jerome H. Friedman. *Multivariate Adaptive Regression Splines (with discussion).* Annals of Statistics 19/1, 1991. `https://statistics.stanford.edu/research/multivariate-adaptive-regression-splines`. Cited on pages 4, 5, and 58.

[8] Jerome H. Friedman. *Fast MARS.* Stanford University Department of Statistics, Technical Report 110, 1993. `https://statistics.stanford.edu/research/fast-mars`. Cited on pages 4, 5, and 11.

[9] Jerome H. Friedman and Bernard W. Silverman. *Flexible Parsimonious Smoothing and Additive Modeling.* Technometrics, Vol. 31, No. 1., 1989. Cited on pages 5 and 58.

[10] F. Harrell. *Regression Modeling Strategies with Applications to Linear Models, Logistic Regression, and Survival Analysis.* Springer, 2001. `http://biostat.mc.vanderbilt.edu/twiki/bin/view/Main/RmS`. Cited on page 44.

[11] Hastie, Tibshirani, and Buja. *Flexible Discriminant Analysis by Optimal Scoring.* JASA, 1994. `http://www-stat.stanford.edu/~hastie/Papers/fda.pdf`. Cited on pages 10 and 29.

[12] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd Edition).* Springer, 2009. Downloadable from `http://web.stanford.edu/~hastie/ElemStatLearn`. Cited on pages 5, 29, 45, 47, 48, 49, 51, and 58.

[13] Trevor Hastie and Robert Tibshirani. *mda: Mixture and flexible discriminant analysis*, 2011. R package, `https://CRAN.R-project.org/package=mda`. Cited on pages 6, 11, and 27.

[14] Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, and Allan Engelhardt. *caret: Classification and Regression Training*, 2011. R package, `https://CRAN.R-project.org/package=caret`. Cited on pages 56 and 66.

[15] J.R. Leathwick, D. Rowe, J. Richardson, J. Elith, and T. Hastie. *Using Multivariate Adaptive Regression Splines to Predict the Distributions of New Zealand's Freshwater Diadromous Fish*. Freshwater Biology, 2005. `http://www-stat.stanford.edu/~hastie/pub.htm,http://www.botany.unimelb.edu.au/envisci/about/staff/elith.html`. Cited on page 13.

[16] Thomas Lumley using Fortran code by Alan Miller. *leaps: regression subset selection*, 2009. R package, `https://CRAN.R-project.org/package=leaps`. Cited on page 5.

[17] S. Milborrow. *Variance models in earth*, 2015. Vignette for R package earth, `http://www.milbo.org/doc/earth-varmod.pdf`. Cited on pages 4 and 34.

[18] S. Milborrow. *plotmo: Plot a Model's Residuals, Response, and Partial Dependence Plots*, 2018. R package, `https://CRAN.R-project.org/package=plotmo`. Cited on page 9.

[19] S. Milborrow. Derived from mda:mars by T. Hastie and R. Tibshirani. *earth: Multivariate Adaptive Regression Splines*, 2011. R package, `http://www.milbo.users.sonic.net/earth`. Cited on page 4.

[20] Alan Miller. *Subset Selection in Regression (2nd ed.)*. CRC, 2002. `https://jblevins.org/mirror/amiller`. Cited on page 5.

[21] J. Pearce and S. Ferrier. *Evaluating the Predictive Performance of Habitat Models developed using Logistic Regression*. Ecological modelling, 2000. Cited on page 44.

[22] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2014. `http://www.R-project.org`. Cited on page 4.

[23] J.O. Ramsay, Hadley Wickham, Spencer Graves, and Giles Hooker. *fda: Functional Data Analysis*, 2011. R package, `https://CRAN.R-project.org/package=fda`. Cited on page 30.

[24] Wikipedia. *Multivariate Adaptive Regression Splines*. `http://en.wikipedia.org/wiki/Multivariate_adaptive_regression_splines`, Accessed Jan 2016. Cited on page 5.