Package ‘plotmo’

February 15, 2012

Version 1.3-1

Title Plot a model’s response while varying the values of the predictors.

Author Stephen Milborrow

Maintainer Stephen Milborrow <milbo@sonic.net>

Description Plot a model’s response when varying one or two predictors
while holding the other predictors constant. A poor man’s partial dependence plot.

Imports rpart

License GPL-3

URL http://www.milbo.users.sonic.net

Repository CRAN

Date/Publication 2011-09-16 20:09:19

R topics documented:

`plotmo` *Plot a model’s response over a range of predictor values*

Description

Plot a model’s response when varying one or two predictors while holding the other predictors constant. A poor man’s partial dependence plot. See the “Details” section for an overview.

Usage

```
plotmo(object = stop("no 'object' arg"),
       type=NULL, nresponse = NA, clip = TRUE, ylim = NULL,
       center = FALSE, ndiscrete = 5,
       degree1 = TRUE, all1=FALSE, degree2 = TRUE, all2=FALSE,
       grid.func = median, grid.levels = NULL,
       col.response = 0, cex.response = 1, pch.response = 1,
       jitter.response=0, npoints = -1,
       inverse.func = NULL, trace = FALSE,
       nrug = 0, col.degree1 = 1, lty.degree1 = 1, lwd.degree1 = 1,
       col.smooth = 0, lty.smooth = 1, lwd.smooth = 1,
       se = 0, col.shade = "lightgray", col.se = 0, lty.se = 2,
       func = NULL, col.func = "lightblue", lty.func = 1, lwd.func = 1,
       ngrid1 = 50,  grid=FALSE,
       type2 = "persp", ngrid2 = 20,
       col.image = gray(0:10/10), col.persp = "lightblue",
       theta = NA, phi = 30, dvalue = 1, shade = 0.5,
       do.par = TRUE, caption = NULL, main = NULL,
       xlab = "", ylab = "", cex = NULL, cex.lab = 1,
       xflip = FALSE, yflip = FALSE, swapxy = FALSE, ...)
```

Arguments

	To start off, look at the arguments `object`, `type`, `clip`, and `col.response`.
`object`	Model object.
`type`	Type parameter passed to `predict`. For legal values see the `predict` method for your `object` (such as `predict.earth` and `predict.rpart`). By default, `plotmo` tries to automatically select a suitable value (usually `"response"`; if not it will be printed in the caption).
`nresponse`	Which column to use when `predict` returns multiple columns. This can be a column index or column name (which may be abbreviated, partial matching is used). Ignored when `predict` returns a single column.
`clip`	Default is `TRUE`, meaning plot only predicted values that are in the expected range. Use `FALSE` to plot all values. See “The `clip` argument” section below for details.
`ylim`	Three possibilities: (i) `NULL` (default) all y axes have same limits (where “y” is actually “z” on degree2 plots). The limits are the min and max values of the predicted response across all plots (after applying `clip`). (ii) `NA` each graph has its own y limits. (iii) `c(ymin,ymax)` graphs have the specified y limits.
`center`	Center the plotted response. Default is `FALSE`. (This is an initial implementation of centering and will change.)

`ndiscrete`
: Default `5` (a somewhat arbitrary value). Variables with no more than `ndiscrete` unique values are plotted as quantized in plots (a staircase rather than a curve). Factors are always considered discrete.

`degree1`
: Index vector specifying which main effect plots to include. Default is `TRUE`, meaning all degree1 plots (the `TRUE` gets recycled). Use `FALSE` (or `0`) for no degree1 plots.
Note that this indexes `plotmo` plots, not columns in `x`. Probably the easiest way to use this argument (and `degree2`) is to first use the default (and possibly `all1=TRUE`) to plot all figures. This shows how the figures are numbered. Then replot using `degree1` to select the figures you want, for example, `degree1=c(1,3)`.
New in version 1.3-0: `degree1` may be a character vector specifying which variables to plot e.g. `degree1=c("wind", "vis")`. Note: `grep` is used for matching. Thus `"wind"` will match all variables that have `"wind"` in their names. Use `"^wind$"` to match only the variable named `"wind"`.

`all1`
: Default is `FALSE`. Use `TRUE` to plot all predictors, not just those usually selected by `plotmo`. See “Which variables are plotted?” below. The `all1` argument increases the number of plots; the `degree1` argument reduces the number of plots.

`degree2`
: Index vector specifying which interaction plots to include. Default is `TRUE`, meaning all degree2 plots.
New in version 1.3-0: `degree2` may be a character vector specifying which variables to plot (`grep` is used for matching).

`all2`
: Default is `FALSE`. Use `TRUE` to plot all pairs of predictors, not just those usually selected by `plotmo`.

`grid.func`
: Function applied to columns of the `x` matrix to fix the values of variables not on the axes. Default is `median`. (This argument is ignored for factors. The first level of factors is used. That can be changed with `grid.levels`.) Example:

```
grid.func <- function(x) quantile(x)[2] # 25% quantile
plotmo(fit, grid.func = grid.func)
```

`grid.levels`
: Default is `NULL`. Else a list of variables and their fixed value to be used when the variable is not on the axis. Supersedes `grid.func` for variables in the list. Names and values can be abbreviated, partial matching is used. Example:
`plotmo(fit, grid.levels=list(sex="m", age=21))`.

`col.response`
: Color of response points (or response sites in degree2 plots). This refers to the response `y` in the data used to build the model. Default is 0, don't plot the response. Can be a vector, for example,
`col.response=as.numeric(survived)+2`.

`cex.response`
: Relative size of response points. Default is `1`. Applies only if `col.response!=0`.

`pch.response`
: Plot character for response points. Default is `1`. Applies only if `col.response!=0`.

`jitter.response`
: Amount to jitter the response points. Applies only if `col.response!=0`. Default `0`, no jitter. A typical useful value is `.3`, but it depends on the data. Points are jittered horizontally and vertically. Note: the points for factors and discrete

	variables and responses are always jittered (because unambiguous space is available), even when `jitter.response` is zero.
`npoints`	Number of response points to be plotted. Applies only if `col.response!=0`. Default is the special value `-1` meaning all. Otherwise a sample of `npoints` points is taken.
`inverse.func`	Default is `NULL`. Else a function applied to the predicted response before plotting. For example, you could use `inverse.func=exp` if your model formula is `log(y)~x`.
`trace`	Default is `FALSE`. Use `TRUE` to trace operation. Use values greater than `1` for more detailed tracing.

The following arguments are for degree1 (main effect) plots

`nrug`	Number of points in (jittered) rug. Default is `0`, no rug. Special value `-1` for all. Otherwise a sample of `nrug` points is taken.
`col.degree1`	Color of degree1 lines. Default is `1`.
`lty.degree1`	Line type of degree1 lines. Default is `1`.
`lwd.degree1`	Line width of degree1 lines. Default is `1`.
`col.smooth`	Color of smoothed line through the response points. (The points themselves will not be plotted unless `col.response` is set.) This refers to the response `y` in the data used to build the model. Default is `0`, no line. Smoothing is done with `lowess`, but for factors and discrete predictors (`<= ndiscrete` levels) the mean response at each level is plotted instead (no smoothing). Example:

```
fit <- earth(O3~., data=ozone1)
plotmo(fit, degree1=c(4,8), col.resp="gray", col.smooth=2)
```

`lty.smooth`	Default is `1`. Applies only if `col.smooth!=0`.
`lwd.smooth`	Default is `1`. Applies only if `col.smooth!=0`.
`se`	Draw standard error bands at plus and minus `se` times the pointwise standard errors. Default is `0`, no standard error bands. A typical value would be `2`. The predict method for the model `object` must support standard errors, i.e. be callable with `se.fit=TRUE` (such as `predict.lm` but not `predict.earth`). Example:

```
fit <- lm(stack.loss~., stackloss)
plotmo(fit, se=2, col.response=2, nrug=-1)
```

`col.shade`	Color of `se` shading. Default is `"lightgray"`. Use `0` for no shading. Applies only if `se!=0`.
`col.se`	Color of `se` lines. Default is `0`, no lines just shading. Applies only if `se!=0`.
`lty.se`	Line type of `se` lines. Default is `2`.
`func`	Superimpose `func(x)` if `func` is not `NULL`. Default is `NULL`. This is useful if you are comparing the model to a known function. The `func` is called for each plot with a single argument which is a data frame with columns in the same order as the predictors in the `formula` or `x` used to build the model. Use `trace=TRUE` to see the column names and first few rows of this dataframe.

`col.func`	Color of `func` line. Default is `"lightblue"`.
`lwd.func`	Line width of `func` line. Default is 1.
`lty.func`	Line type of `func` line. Default is 1.
`ngrid1`	Number of points in degree1 plots. Default is `50`.
`grid`	Default `FALSE`. Use `TRUE` to add a `grid` to the degree1 plots. You can also specify a color here, e.g. `grid="darkgray"`.
`xlab`	Horizontal axis label on degree1 plots (for degree2 plots the labels are always the predictor names). Default is `""`, no label, which gives more plottable area. The special value `NULL` means use the current variable name as the label. (If you use `NULL`, you may want to use `main=""` to avoid redundant labeling.)
`ylab`	Vertical axis label. Values as for `xlab`.
	The following arguments are for degree2 plots
`type2`	Degree2 plot type. One of `"persp"` (default), `"contour"`, or `"image"`.
`ngrid2`	Grid size for degree2 plots (`ngrid2 x ngrid2` points are plotted, but less for factors and variables with less than `ngrid2` discrete values). Default is `20`. Note 1: the default will often be too small for `contour` and `image` plots. Note 2: with large `ngrid2` values, `persp` plots look better with `border=NA`.
`col.image`	Colors of `image` plot. Default is `gray(0:10/10)`, a range of grays. Clipped values will be displayed in blue (only applies if `clip=TRUE`).
`col.persp`	Color of `persp` surface. Default is `"lightblue"`. Use 0 for no color.
`theta`	Rotation parameter for `persp`. Default is `NA`, meaning automatically rotate each graph so the highest corner is furthest away. Use `trace=TRUE` to see the calculated value for `theta`. Higher values of `theta` rotate clockwise.
`phi`	Passed to `persp`. Default is `30`. Lower values to view from the side; higher to view from above.
`dvalue`	Passed to `persp` as `d`. Default is `1`. The name was changed from `d` to avoid partial matching problems.
`shade`	Passed to `persp`. Default is `0.5`.
	The following are related to `par` and other graphical settings.
`do.par`	Default is `TRUE`, meaning start a new page and call `par` as appropriate (this adjusts `mfrow`, `cex`, `mar`, and `mgp`). Use `FALSE` to use the current graphics settings. The value `2` means act like `TRUE` but do not restore the `par` settings to their original state (useful for slipping in a few more plots on the same page).
`caption`	Overall caption. By default create captions automatically from the `type`, response name, and `call`.
`main`	A vector of titles, one for each plot. By default generate titles automatically from the variable names. See also `caption`, for the overall title.
`cex`	Character expansion.
`cex.lab`	Relative size of axis labels and text. Default `1`.

`xflip`	Default `FALSE`. Use `TRUE` to flip the direction of the x axis. This argument (and `yflip` and `swapxy`) is useful when comparing to a plot from another source and you want the axes to be the same. (Note that `xflip` and `yflip` cannot be used on the `persp` plots, a limitation of the `persp` function.)
`yflip`	Default `FALSE`. Use `TRUE` to flip the direction of the y axis of the degree2 graphs.
`swapxy`	Default `FALSE`. Use `TRUE` to swap the x and y axes on the degree2 graphs.
`...`	Extra arguments are passed on to the plotting functions. (For `persp` plots, `ticktype="d", nticks=2` is useful.)

Details

`Plotmo` can be used on a wide variety of regression models. It plots a degree1 (main effect) plot by calling `predict` to predict the response when changing one variable while holding all other variables at their median values. For degree2 (interaction) plots, two variables are changed while holding others at their medians. The first level is used instead of the median for factors. You can change this with the `grid.func` and `grid.levels` arguments.

Each graph shows only a thin slice of the data because most variables are fixed. Please be aware of that when interpreting the graph — over-interpretation is a temptation.

`Plotmo` was originally part of the `earth` package and a few connections to that package still remain.

Limitations

NAs are not yet supported. To prevent confusing error messages from functions called by `plotmo`, it is safest to remove NAs before building your model. (However, `rpart` models are treated specially by `plotmo`. For these, `plotmo` predicts with `na.pass` so `plotmo` can be used with `rpart`'s default NA handling.)

Keep the variable names in the original model formula simple. Use temporary variables or `attach` rather than using `$` and similar in formulas.

`Plotmo` evaluates the model data in the `environment` used when the model was built, if that environment was saved with the model (typically this is the case if the formula interface was used to the model function). If the environment was not saved with the model (typically if the `x,y` interface was used), the model data is evaluated in the environment in which `plotmo` is called.

Alternatives

An alternative approach is to use partial-dependence plots (e.g. *The Elements of Statistical Learning* 10.13.2). `Plotmo` sets the "other" variables to their median value, whereas in a partial-dependence plot at each plotted point the effect of the other variables is averaged. In general, partial-dependence plots and `plotmo` plots will differ, but for additive models the *shape* of the curves will match identically. Eventually `plotmo` will be enhanced to draw partial-dependence plots.

`Termplot` is effective but can be used only on models with a `predict` method that supports `type="terms"`, and it does not generate degree2 plots.

Some other possibilites for plotting the response on a per-predictor basis are partial-residual plots, partial-regression variable plots, and marginal-model plots (e.g. `crPlots`, `avPlots`, and `marginalModelPlot` in the `car-package`). These plots are orientated towards linear models.

Which variables are plotted?

The set of variables plotted for some common objects is listed below. This may leave out variables that you would like to see — in that case use `all1=TRUE` and `all2=TRUE`.

- **o** `earth` `degree1`: variables in additive (non interaction) terms
  `degree2`: variables appearing together in interaction terms.
- **o** `rpart` `degree1`: variables used in the tree
  `degree2`: parent-child pairs.
- **o** `randomForest` `degree1`: all variables
  `degree2`: pairs of the four most important variables (ranked on the first column of `object$importance`).
- **o** `gbm` `degree1`: variables with `relative.influence >= 1%`
  `degree2`: pairs of the four variables with the largest relative influence.
- **o** `lm, glm, gam, lda`**, etc. are processed using** `plotmo`**'s default methods:** `degree1`: all variables
  `degree2`: variables in formula terms like `x1*x2`, `x1:x2` and `s(x1,x2)`.

The `clip` argument

With the default `clip=TRUE`, predicted values out of the expected range are not displayed.

Generally, the "expected range" is the range of the response `y` used when building the model. But that depends on the type of model, and `plotmo` knows about some special cases. For example, it knows that for some models we are predicting a probability, and it scales the axes accordingly, `0` to `1`. However, `plotmo` cannot know about every possible model and prediction `type`, and will sometimes determine the expected response range incorrectly. In that case use `clip=FALSE`.

The default `clip` is `TRUE` because it is a useful sanity check to test that the predicted values are in the expected range. While not necessarily an error, predictions outside the expected range are usually something we want to know about. Also, with `clip=FALSE`, a few errant predictions can expand the entire y-axis, making it difficult to see the shape of the other predictions.

Using `plotmo` on various models

Here are some examples which illustrate `plotmo` on various objects. (The models here are just for illustrating `plotmo` and shouldn't be taken too seriously.)

```
# use a small set of variables for illustration
library(earth) # for ozone1 data
data(ozone1)
oz <- ozone1[, c("O3", "humidity", "temp", "ibt")]

lm.model <- lm(O3 ~ humidity + temp*ibt, data=oz)       # linear model
plotmo(lm.model, se=2, col.response="gray", nrug=-1)

library(rpart)                                          # rpart
rpart.model <- rpart(O3 ~ ., data=oz)
plotmo(rpart.model, all2=TRUE)

library(randomForest)                                   # randomForest
rf.model <- randomForest(O3~., data=oz)
plotmo(rf.model)
# partialPlot(rf.model, oz, temp) # compare to partial-dependence plot

library(gbm)                                            # gbm
gbm.model <- gbm(O3~., data=oz, dist="gaussian", inter=2, n.trees=1000)
```

```
plotmo(gbm.model)
# plot(gbm.model, i.var=2) # compare to partial-dependence plots
# plot(gbm.model, i.var=c(2,3))

library(mgcv)                                           # gam
gam.model <- gam(O3 ~ s(humidity)+s(temp)+s(ibt)+s(temp,ibt), data=oz)
plotmo(gam.model, se=2, all2=TRUE)

library(nnet)                                           # nnet
set.seed(4)
nnet.model <- nnet(O3~., data=scale(oz), size=2, decay=0.01, trace=FALSE)
plotmo(nnet.model, type="raw", all2=T)

library(MASS)                                           # qda
lcush <- data.frame(Type=as.numeric(Cushings$Type),log(Cushings[,1:2]))
lcush <- lcush[1:21,]
qda.model <- qda(Type~., data=lcush)
plotmo(qda.model, type="class", all2=TRUE,
   type2="contour", ngrid2=100, nlevels=2, drawlabels=FALSE,
   col.response=as.numeric(lcush$Type)+1,
   pch.response=as.character(lcush$Type))
```

Extending plotmo

`Plotmo` needs to access the data used to build the model. It does that with the method functions listed below. The default methods suffice for many objects. However, the default methods don't work (`plotmo` will issue an error message) if the model function did not save the call or data with the object in a standard fashion. Object-specific methods can usually be written to deal with such issues. See `plotmo.methods.gbm.R` in the `plotmo` source code for an example. The methods are:

`plotmo.prolog` called before plotting begins, sanity check of the `object`

`plotmo.predict` invokes `predict` for each sub-plot

`get.plotmo.x` the model matrix `x`

`get.plotmo.y` the model response `y`

`get.plotmo.default.type` the value of the `type` argument when not specified by the user

`get.plotmo.singles` the vector of variables to be plotted in degree1 plots

`get.plotmo.pairs` the array of pairs to be plotted in degree2 plots

`get.plotmo.ylim` the value of `ylim` when not specified by the user

`get.plotmo.clip.limits` the clip range when `clip=TRUE`

Common error messages

o `Error in match.arg(type): 'arg' should be one of ...`

The message is probably issued by the `predict` method for your model `object`. Set `type` to a legal value as described on the help page for the `predict` method for your object.

o `Error: predicted values are out of ylim, try clip=FALSE`

Probably `plotmo` has incorrectly determined the expected range of the response, and hence also `ylim`. Re-invoke `plotmo` with `clip=FALSE`. See the section “The `clip` argument”.

o `Error: predict.lm(xgrid, type="response") returned the wrong length`

o `Warning: 'newdata' had 100 rows but variable(s) found have 30 rows`

o `Error: variable 'x' was fitted with type "nmatrix.2" but type "numeric" was supplied`

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

These and similar messages usually mean that `predict` is misinterpreting the new data generated by `plotmo`.

The underlying issue is that many `predict` methods, including `predict.lm`, seem to reject any reasonably constructed new data for certain models. The work-around is to simplify or standardize the way the model function is called. Use a formula and a data frame, or at least explicitly name the the variables rather than passing a matrix. Use simple variable names (so `x1` rather than `dat$x1`, for example).

If the symptoms persist after changing the way the model is called, and the model is not one of those listed in “Which variables are plotted”, it is possible that the model class is not supported by `plotmo`. See “Extending plotmo”.

o `Error: get.plotmo.x.default cannot get the x matrix`

This and similar messages mean that `plotmo` cannot get the data it needs from the model `object`.

You can try simplifying and standardizing the way the model function is called, as described above. Perhaps you need to use `keepxy` or similar in the call to the model function, so the data is attached to the object and available for `plotmo`. Is a variable that was used to build the model no longer available in the environment when `plotmo` is called?

o `Error: this object is not supported by plotmo`

`Plotmo`’s default methods are insufficient for your model object. See “Extending plotmo” above (and contact the author — this is often easy to fix).

FAQ

o *I am not seeing any interaction plots. How can I change that?*

Use `all2=TRUE`. By default, degree2 plots are drawn only for some types of model. See the section “Which variables are plotted?”.

o *The* `persp` *display is unnaturally jagged. How can I change that?*

Use `clip=FALSE`. The jaggedness is probably an artifact of the way `persp` works at the boundaries. You can also try increasing `ngrid2`.

o *The* `image` *display has blue “holes” in it. What gives?*

The holes are areas where the predicted response is out-of-range. Try using `clip=FALSE`.

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

Use `do.par=FALSE` or `do.par=2`. With the default `do.par=TRUE`, `plotmo` restores the `par` parameters and axis scales to their values before `plotmo` was called.

Author(s)

Stephen Milborrow

See Also

There is section on `plotmo` in the `rpart.plot` vignette "Plotting `rpart` trees with `prp`".

Examples

```
library(rpart)
data(kyphosis)
rpart.model <- rpart(Kyphosis~., data=kyphosis)
plotmo(rpart.model, type="prob", nresponse="present")
```

`plotmo.methods` *Please ignore*

Description

Methods exported for use by `earth`.

Usage

```
get.plotmo.singles(object, env, x, trace, all1)
get.plotmo.pairs(object, env, x, trace, all2)
get.plotmo.y(object, env, y.column, expected.len, trace)
```

Arguments

`all1,all2,env,expected.len`	See `plotmo.methods.R`.
`object,trace,x,y,y.column`	See `plotmo.methods.R`.

Index