Package: plotmo (via r-universe)

September 1, 2024

Version 3.6.4

Title Plot a Model's Residuals, Response, and Partial Dependence Plots

Maintainer Stephen Milborrow <milbo@sonic.net>

Depends R (>= 3.4.0), Formula (>= 1.2-3), plotrix

Description Plot model surfaces for a wide variety of models using partial dependence plots and other techniques. Also plot model residuals and other information on the model.

Suggests C50 (>= 0.1.0-24), earth (>= 5.1.2), gbm (>= 2.1.1), glmnet (>= 2.0.5), glmnetUtils (>= 1.0.3), MASS (>= 7.3-51), mlr (>= 2.12.1), neuralnet (>= 1.33), partykit (>= 1.2-2), pre (>= 0.5.0), rpart (>= 4.1-15), rpart.plot (>= 3.0.8)

License GPL-3

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

NeedsCompilation no

Author Stephen Milborrow [aut, cre]

Date/Publication 2024-08-31 03:10:02 UTC

Repository https://stephenmilborrow.r-universe.dev

RemoteUrl https://github.com/cran/plotmo

RemoteRef HEAD

RemoteSha 4bae0a435d023abe8fbca51f5e370ebbfa95a39c

Contents

plotmo *Plot a model's response over a range of predictor values (the model surface)*

Description

Plot model surfaces for a wide variety of models.

This function plots the model's response when varying one or two predictors while holding the other predictors constant (a poor man's partial-dependence plot).

It can also generate partial-dependence plots (by specifying `pmethod="partdep"`).

Please see the plotmo vignette (also available here).

Usage

```
plotmo(object=stop("no 'object' argument"),
    type=NULL, nresponse=NA, pmethod="plotmo",
    pt.col=0, jitter=.5, smooth.col=0, level=0,
    func=NULL, inverse.func=NULL, nrug=0, grid.col=0,
    type2="persp",
    degree1=TRUE, all1=FALSE, degree2=TRUE, all2=FALSE,
    do.par=TRUE, clip=TRUE, ylim=NULL, caption=NULL, trace=0,
    grid.func=NULL, grid.levels=NULL, extend=0,
    ngrid1=50, ngrid2=20, ndiscrete=5, npoints=3000,
    center=FALSE, xflip=FALSE, yflip=FALSE, swapxy=FALSE, int.only.ok=TRUE,
    ...)
```

Arguments

`object`	The model object.
`type`	Type parameter passed to `predict`. For allowed values see the `predict` method for your `object` (such as `predict.earth`). By default, `plotmo` tries to automatically select a suitable value for the model in question (usually `"response"`) but this will not always be correct. Use `trace=1` to see the `type` argument passed to `predict`.
`nresponse`	Which column to use when `predict` returns multiple columns. This can be a column index, or a column name if the `predict` method for the model returns column names. The column name may be abbreviated, partial matching is used.
`pmethod`	Plotting method. One of: `"plotmo"` (default) Classic plotmo plots i.e. the background variables are fixed at their medians (or first level for factors). `"partdep"` Partial dependence plots, i.e. at each point the effect of the background variables is averaged. `"apartdep"` Approximate partial dependence plots. Faster than `"partdep"` especially for big datasets. Like `"partdep"` but the background variables are averaged over a subset of `ngrid1` cases (default 50), rather than all cases in the

training data. The subset is created by selecting rows at equally spaced intervals from the training data after sorting the data on the response values (ties are randomly broken). The same background subset of `ngrid1` cases is used for both degree1 and degree2 plots.

`pt.col` The color of response points (or response sites in degree2 plots). This refers to the response `y` in the data used to build the model. Note that the displayed points are jittered by default (see the `jitter` argument).
Default is `0`, display no response points.
This can be a vector, like all such arguments – for example `pt.col = as.numeric(survived)+2` to color points by their survival class.
You can modify the plotted points with `pt.pch`, `pt.cex`, etc. (these get passed via `plotmo`'s "`...`" argument). For example, `pt.cex = weights` to size points by their weight. To label the points, set `pt.pch` to a character vector.

`jitter` Applies only if `pt.col` is specified.
The default is `jitter=.5`, automatically apply some jitter to the points. Points are jittered horizontally and vertically.
Use `jitter=0` to disable this automatic jittering. Otherwise something like `jitter=1`, but the optimum value is data dependent.

`smooth.col` Color of smooth line through the response points. (The points themselves will not be plotted unless `pt.col` is specified.) Default is `0`, no smooth line.
Example:

```
mod <- lm(Volume~Height, data=trees)
plotmo(mod, pt.color=1, smooth.col=2)
```

You can adjust the amount of smoothing with `smooth.f`. This gets passed as `f` to `lowess`. The default is `.5`. Lower values make the line more wiggly.

`level` Draw estimated confidence or prediction interval bands at the given `level`, if the predict method for the model supports them.
Default is `0`, bands not plotted. Else a fraction, for example `level=.95`. See "*Prediction intervals*" in the `plotmo` vignette. Example:

```
mod <- lm(log(Volume)~log(Girth), data=trees)
plotmo(mod, level=.95)
```

You can modify the color of the bands with `level.shade` and `level.shade2`.

`func` Superimpose `func(x)` on the plot. Example:

```
mod <- lm(Volume~Girth, data=trees)
estimated.volume <- function(x) .17 * x$Girth^2
plotmo(mod, pt.col=2, func=estimated.volume)
```

The `func` is called for each plot with a single argument which is a dataframe with columns in the same order as the predictors in the `formula` or `x` used to build the model. Use `trace=2` to see the column names and first few rows of this dataframe.

`inverse.func` A function applied to the response before plotting. Useful to transform a transformed response back to the original scale. Example:

```
mod <- lm(log(Volume)~., data=trees)
plotmo(mod, inverse.func=exp)    # exp() is inverse of log()
```

`nrug` Number of ticks in the rug along the bottom of the plot
Default is `0`, no rug.
Use `nrug=TRUE` for all the points.
Else specify the number of quantiles e.g. use `nrug=10` for ticks at the 0, 10, 20, ..., 100 percentiles.
Modify the rug ticks with `rug.col`, `rug.lwd`, etc.
The special value `nrug="density"` means plot the density of the points along the bottom. Modify the density plot with `density.adjust` (default is `.5`), `density.col`, `density.lty`, etc.

`grid.col` Default is `0`, no grid. Else add a background grid of the specified color to the degree1 plots. The special value `grid.col=TRUE` is treated as `"lightgray"`.

`type2` Degree2 plot type. One of `"persp"` (default), `"image"`, or `"contour"`. You can pass arguments to these functions if necessary by using `persp.`, `image.`, or `contour.` as a prefix. Examples:

```
plotmo(mod, persp.ticktype="detailed", persp.nticks=3)
plotmo(mod, type2="image")
plotmo(mod, type2="image", image.col=heat.colors(12))
plotmo(mod, type2="contour", contour.col=2, contour.labcex=.4)
```

`degree1` An index vector specifying which subset of degree1 (main effect) plots to include (after selecting the relevant predictors as described in "*Which variables are plotted?*" in the `plotmo` vignette).
Default is `TRUE`, meaning all (the `TRUE` gets recycled). To plot only the third plot use `degree1=3`. For no degree1 plots use `degree1=0`.

Note that `degree1` indexes plots on the page, not columns of `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(1,3,4)`.

Can also be a character vector specifying which variables to plot. Examples:
`degree1="wind"`
`degree1=c("wind", "vis")`.

Variables names are matched with grep. Thus `"wind"` will match all variables with `"wind"` anywhere in their name. 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`.
The `all1` argument increases the number of plots; the `degree1` argument reduces the number of plots.

`degree2` An index vector specifying which subset of degree2 (interaction) plots to include.

Default is TRUE meaning all (after selecting the relevant interaction terms as described in "*Which variables are plotted?*" in the plotmo vignette).

Can also be a character vector specifying which variables to plot (grep is used for matching). Examples:
degree2="wind" plots all degree2 plots for the wind variable.
degree2=c("wind", "vis") plots just the wind:vis plot.

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

do.par — One of NULL, FALSE, TRUE, or 2, as follows:
do.par=NULL. Same as do.par=FALSE if the number of plots is one; else the same as TRUE.
do.par=FALSE. Use the current par settings. You can pass additional graphics parameters in the "..." argument.
do.par=TRUE (default). Start a new page and call par as appropriate to display multiple plots on the same page. This automatically sets parameters like mfrow and mar. You can pass additional graphics parameters in the "..." argument.
do.par=2. Like do.par=TRUE but don't restore the par settings to their original state when plotmo exits, so you can add something to the plot.

clip — The default is clip=TRUE, meaning ignore very outlying predictions when determining the automatic ylim. This keeps ylim fairly compact while still covering all or nearly all the data, even if there are a few crazy predicted values. See "*The* ylim *and* clip *arguments*" in the plotmo vignette.
Use clip=FALSE for no clipping.

ylim — Three possibilities:
ylim=NULL (default). Automatically determine a ylim to use across all graphs.
ylim=NA. Each graph has its own ylim.
ylim=c(ymin,ymax). Use the specified limits across all graphs.

caption — Overall caption. By default create the caption automatically. Use caption="" for no caption. (Use main to set the title of individual plots, can be a vector.)

trace — Default is 0.
trace=1 (or TRUE) for a summary trace (shows how predict is invoked for the current object).
trace=2 for detailed tracing.
trace=-1 inhibits the messages usually issued by plotmo, like the plotmo grid:, calculating partdep, and nothing to plot messages. Error and warning messages will be printed as usual.

grid.func — Function applied to columns of the x matrix to pin the values of variables not on the axis of the current plot (the "background" variables).
The default is a function which for numeric variables returns the median and for logical and factors variables returns the value occurring most often in the training data.
Examples:

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

This argument is not related to the `grid.col` argument.
This argument can be overridden for specific variables—see `grid.levels` below.

`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(mod, grid.levels=list(sex="m", age=21))
```

`extend` Amount to extend the horizontal axis in each plot. The default is `0`, do not extend (i.e. use the range of the variable in the training data). Else something like `extend=.5`, which will extend both the lower and upper `xlim` of each plot by 50%.
This argument is useful if you want to see how the model performs on data that is beyond the training data; for example, you want to see how a time-series model performs on future data.
This argument is currently implemented only for degree1 plots. Factors and discrete variables (see the `ndiscrete` argument) are not extended.

`ngrid1` Number of equally spaced `x` values in each degree1 plot. Default is `50`. Also used as the number of background cases for `pmethod="apartdep"`.

`ngrid2` Grid size for degree2 plots (`ngrid2` x `ngrid2` points are plotted). Default is `20`.
The default will sometimes be too small for `contour` and `image` plots.
With large `ngrid2` values, `persp` plots look better with `persp.border=NA`.

`npoints` Number of response points to be plotted (a sample of `npoints` points is plotted). Applies only if `pt.col` is specified.
The default is 3000 (not all, to avoid overplotting on large models). Use `npoints=TRUE` or `-1` for all points.

`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. Variables with non-integer values are always considered non-discrete.
Use `ndiscrete=0` if you want to plot the response for a variable with just a few integer values as a line or a curve, rather than a staircase.

`int.only.ok` Plot the model even if it is an intercept-only model (no predictors are used in the model). Do this by plotting a single degree1 plot for the first predictor.
The default is `TRUE`. Use `int.only.ok=FALSE` to instead issue an error message for intercept-only models.

`center` Center the plotted response. Default is `FALSE`.

`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.

`...` Dot arguments are passed to the predict and plot functions. Dot argument names, whether prefixed or not, should be specified in full and not abbreviated.

"Prefixed" arguments are passed directly to the associated function. For example the prefixed argument `persp.col="pink"` passes `col="pink"` to `persp()`, overriding the global `col` setting. To send an argument to `predict` whose name may alias with `plotmo`'s arguments, use `predict.` as a prefix. Example:

```
plotmo(mod, s=1)        # error: arg matches multiple formal args
plotmo(mod, predict.s=1)  # ok now: s=1 will be passed to predict()
```

The prefixes recognized by `plotmo` are:

`predict.`	passed to the `predict` method for the model
`degree1.`	modifies degree1 plots e.g. `degree1.col=3, degree1.lwd=2`
`persp.`	arguments passed to `persp`
`contour.`	arguments passed to `contour`
`image.`	arguments passed to `image`
`pt.`	see the `pt.col` argument (arguments passed to `points` and `text`)
`smooth.`	see the `smooth.col` argument (arguments passed to `lines` and `lowess`)
`level.`	see the `level` argument (`level.shade`, `level.shade2`, and arguments for `polygon`)
`func.`	see the `func` argument (arguments passed to `lines`)
`rug.`	see the `nrug` argument (`rug.jitter`, and arguments passed to `rug`)
`density.`	see the `nrug` argument (`density.adjust`, and arguments passed to `lines`)
`grid.`	see the `grid.col` argument (arguments passed to `grid`)
`caption.`	see the `caption` argument (arguments passed to `mtext`)
`par.`	arguments passed to `par` (only necessary if a `par` argument name clashes with a `plotmo` argument)
`prednames.`	Use `prednames.abbreviate=FALSE` for full predictor names in graph axes.

The `cex` argument is relative, so specifying `cex=1` is the same as not specifying `cex`.

For backwards compatibility, some dot arguments are supported but not explicitly documented. For example, the old argument `col.response` is no longer in `plotmo`'s formal argument list, but is still accepted and treated like the new argument `pt.col`.

Note

In general this function won't work on models that don't save the call and data with the model in a standard way. For further discussion please see "*Accessing the model data*" in the plotmo vignette. Package authors may want to look at Guidelines for S3 Regression Models (also available here).

By default, `plotmo` tries to use sensible model-dependent defaults when calling `predict`. Use `trace=1` to see the arguments passed to `predict`. You can change the defaults by using `plotmo`'s

type argument, and by using dot arguments prefixed with predict. (see the description of "...") above).

See Also

Please see the plotmo vignette (also available here).

Examples

```
if (require(rpart)) {
    data(kyphosis)
    rpart.model <- rpart(Kyphosis~., data=kyphosis)
    # pass type="prob" to plotmo's internal calls to predict.rpart, and
    # select the column named "present" from the matrix returned by predict.rpart
    plotmo(rpart.model, type="prob", nresponse="present")
}
if (require(earth)) {
    data(ozone1)
    earth.model <- earth(O3 ~ ., data=ozone1, degree=2)
    plotmo(earth.model)
    # plotmo(earth.model, pmethod="partdep") # partial dependence plots
}
```

---

plotmo.misc *Ignore*

Description

Miscellaneous functions exported for internal use by earth and other packages. You can ignore these.

Usage

```
# for earth
plotmo_fitted(object, trace, nresponse, type, ...)
plotmo_cum(rinfo, info, nfigs=1, add=FALSE,
           cum.col1, grid.col, jitter=0, cum.grid="percentages", ...)
plotmo_nresponse(y, object, nresponse, trace, fname, type="response")
plotmo_rinfo(object, type=NULL, residtype=type, nresponse=1,
    standardize=FALSE, delever=FALSE, trace=0,
    leverage.msg="returned as NA", expected.levs=NULL, labels.id=NULL, ...)
plotmo_predict(object, newdata, nresponse,
    type, expected.levs, trace, inverse.func=NULL, ...)
plotmo_prolog(object, object.name, trace, ...)
plotmo_resplevs(object, plotmo_fitted, yfull, trace)
plotmo_rsq(object, newdata, trace=0, nresponse=NA, type=NULL, ...)
plotmo_standardizescale(object)
plotmo_type(object, trace, fname="plotmo", type, ...)
```

```
plotmo_y(object, nresponse=NULL, trace=0, expected.len=NULL,
    resp.levs=NULL, convert.glm.response=!is.null(nresponse))
## Default S3 method:
plotmo.pairs(object, x, nresponse, trace, all2, ...)
## Default S3 method:
plotmo.singles(object, x, nresponse, trace, all1, ...)
## Default S3 method:
plotmo.y(object, trace, naked, expected.len, ...)
# plotmo methods
plotmo.convert.na.nresponse(object, nresponse, yhat, type="response", ...)
plotmo.pairs(object, x, nresponse, trace, all2, ...)
plotmo.pint(object, newdata, type, level, trace, ...)
plotmo.predict(object, newdata, type, ..., TRACE)
plotmo.prolog(object, object.name, trace, ...)
plotmo.residtype(object, ..., TRACE)
plotmo.singles(object, x, nresponse, trace, all1, ...)
plotmo.type(object, ..., TRACE)
plotmo.x(object, trace, ...)
plotmo.y(object, trace, naked, expected.len, nresponse=1, ...)
```

Arguments

```
...             -
add             -
all1            -
all2            -
convert.glm.response
                -
cum.col1        -
cum.grid        -
delever         -
expected.len    -
expected.levs   -
fname           -
grid.col        -
info            -
inverse.func    -
jitter          -
labels.id       -
level           -
leverage.msg    -
naked           -
newdata         -
```

nfigs	-
nresponse	-
object.name	-
object	-
plotmo_fitted	-
residtype	-
resp.levs	-
rinfo	-
standardize	-
TRACE	-
trace	-
type	-
x	-
yfull	-
yhat	-
y	-

`plotres` *Plot the residuals of a regression model*

Description

Plot the residuals of a regression model.

Please see the plotres vignette (also available here).

Usage

```
plotres(object = stop("no 'object' argument"),
    which = 1:4, info = FALSE, versus = 1,
    standardize = FALSE, delever = FALSE, level = 0,
    id.n = 3, labels.id = NULL, smooth.col = 2,
    grid.col = 0, jitter = 0,
    do.par = NULL, caption = NULL, trace = 0,
    npoints = 3000, center = TRUE,
    type = NULL, nresponse = NA,
    object.name = quote.deparse(substitute(object)), ...)
```

Arguments

object	The model object.
which	Which plots do draw. Default is 1:4. 1 Model plot. What gets plotted here depends on the model class. For example, for earth models this is a model selection plot. Nothing will be displayed for some models. For details, please see the plotres vignette. 2 Cumulative distribution of abs residuals 3 Residuals vs fitted 4 QQ plot 5 Abs residuals vs fitted 6 Sqrt abs residuals vs fitted 7 Abs residuals vs log fitted 8 Cube root of the squared residuals vs log fitted 9 Log abs residuals vs log fitted
info	Default is FALSE. Use TRUE to print extra information as follows: i) Display the distribution of the residuals along the bottom of the plot. ii) Display the training R-Squared. iii) Display the Spearman Rank Correlation of the absolute residuals with the fitted values. Actually, correlation is measured against the absolute values of whatever is on the horizontal axis — by default this is the fitted response, but may be something else if the versus argument is used. iv) In the Cumulative Distribution plot (which=2), display additional information on the quantiles. v) Only for which=5 or 9. Regress the absolute residuals against the fitted values and display the regression slope. Robust linear regression is used via rlm in the MASS package. vi) Add various annotations to the other plots.
versus	What do we plot the residuals against? One of: 1 Default. Plot the residuals versus the fitted values (or the log values when which=7 to 9). 2 Residuals versus observation number, after observations have been sorted on the fitted value. Same as versus=1, except that the residuals are spaced uniformly along the horizontal axis. 3 Residuals versus the response. 4 Residuals versus the hat leverages. "b:" Residuals versus the basis functions. Currently only supported for earth, mda::mars, and gam::gam models. A optional regex can follow the "b:" to specify a subset of the terms, e.g. versus="b:wind" will plot terms with "wind" in their name.

Else a character vector specifying which predictors to plot against.
Example 1: `versus=""` plots against all predictors (since the regex `versus=""` matches anything).
Example 2: `versus=c("wind", "vis")` plots predictors with `wind` or `vis` in their name.
Example 3: `versus=c("wind|vis")` equivalent to the above.
Note: These are regexs. Thus `versus="wind"` will match all variables that have `"wind"` in their names. Use `"^wind$"` to match only the variable named `"wind"`.

`standardize` Default is `FALSE`. Use `TRUE` to standardize the residuals. Only supported for some models, an error message will be issued otherwise.
Each residual is divided by by `se_i * sqrt(1 - h_ii)`, where `se_i` is the standard error of prediction and `h_ii` is the leverage (the diagonal entry of the hat matrix). When the variance model holds, the standardized residuals are homoscedastic with unity variance.
The leverages are obtained using `hatvalues`. (For `earth` models the leverages are for the linear regression of the response on the basis matrix `bx`.) A standardized residual with a leverage of 1 is plotted as a star on the axis.
This argument applies to all plots where the residuals are used (including the cumulative distribution and QQ plots, and to annotations displayed by the `info` argument).

`delever` Default is `FALSE`. Use `TRUE` to "de-lever" the residuals. Only supported for some models, an error message will be issued otherwise.
Each residual is divided by `sqrt(1 - h_ii)`. See the `standardize` argument for details.

`level` Draw estimated confidence or prediction interval bands at the given `level`, if the model supports them.
Default is `0`, bands not plotted. Else a fraction, for example `level=0.90`. Example:

```
mod <- lm(log(Volume)~log(Girth), data=trees)
plotres(mod, level=.90)
```

You can modify the color of the bands with `level.shade` and `level.shade2`.
See also "*Prediction intervals*" in the plotmo vignette (but note that `plotmo` needs prediction intervals on *new* data, whereas `plotres` requires only that the model supports prediction intervals on the training data).

`id.n` The largest `id.n` residuals will be labeled in the plot. Default is `3`. Special values `TRUE` and `-1` or mean all.
If `id.n` is negative (but not `-1`) the `id.n` most positive and most negative residuals will be labeled in the plot.
A current implementation restriction is that `id.n` is ignored when there are more than ten thousand cases.

`labels.id` Residual labels. Only used if `id.n > 0`. Default is the case names, or the case numbers if the cases are unnamed.

smooth.col — Color of the smooth line through the residual points. Default is 2, red. Use smooth.col=0 for no smooth line.
You can adjust the amount of smoothing with smooth.f. This gets passed as f to lowess. The default is 2/3. Lower values make the line more wiggly.

grid.col — Default is 0, no grid. Else add a background grid of the specified color to the degree1 plots. The special value grid.col=TRUE is treated as "lightgray".

jitter — Default is 0, no jitter. Passed as factor to jitter to jitter the plotted points horizontally and vertically. Useful for discrete variables and responses, where the residual points tend to be overlaid.

do.par — One of NULL, FALSE, TRUE, or 2, as follows:

do.par=NULL (default). Same as do.par=FALSE if the number of plots is one; else the same as TRUE.

do.par=FALSE. Use the current par settings. You can pass additional graphics parameters in the "..." argument.

do.par=TRUE. Start a new page and call par as appropriate to display multiple plots on the same page. This automatically sets parameters like mfrow and mar. You can pass additional graphics parameters in the "..." argument.

do.par=2. Like do.par=TRUE but don't restore the par settings to their original state when plotres exits, so you can add something to the plot.

caption — Overall caption. By default create the caption automatically. Use caption="" for no caption. (Use main to set the title of an individual plot.)

trace — Default is 0.
trace=1 (or TRUE) for a summary trace (shows how predict and friends are invoked for the model).
trace=2 for detailed tracing.

npoints — Number of points to be plotted. A sample of npoints is taken; the sample includes the biggest twenty or so residuals.
The default is 3000 (not all, to avoid overplotting on large models). Use npoints=TRUE or -1 for all points.

center — Default is TRUE, meaning center the horizontal axis in the residuals plot, so asymmetry in the residual distribution is more obvious.

type — Type parameter passed first to residuals and if that fails to predict. For allowed values see the residuals and predict methods for your object (such as residuals.rpart or predict.earth). By default, plotres tries to automatically select a suitable value for the model in question (usually "response"), but this will not always be correct. Use trace=1 to see the type argument passed to residuals and predict.

nresponse — Which column to use when residuals or predict returns multiple columns. This can be a column index or column name (which may be abbreviated, partial matching is used).

object.name — The name of the object for error and trace messages. Used internally by plot.earth.

`...` Dot arguments are passed to the plot functions. Dot argument names, whether prefixed or not, should be specified in full and not abbreviated.

"Prefixed" arguments are passed directly to the associated function. For example the prefixed argument `pt.col="pink"` passes `col="pink"` to `points()`, overriding the global `col` setting. The prefixes recognized by `plotres` are:

`residuals.`	passed to `residuals`
`predict.`	passed to `predict` (`predict` is called if the call to `residuals` fails)
`w1.`	sent to the model-dependent plot for `which=1` e.g. `w1.col=2`
`pt.`	modify the displayed points e.g. `pt.col=as.numeric(survived)+2` or `pt.cex=.8`.
`smooth.`	modify the smooth line e.g. `smooth.col=0` or `smooth.f=.5`.
`level.`	modify the interval bands, e.g. `level.shade="gray"` or `level.shade2="lightblue"`
`legend.`	modify the displayed `legend` e.g. `legend.cex=.9`
`cum.`	modify the Cumulative Distribution plot (arguments for `plot.stepfun`)
`qq.`	modify the QQ plot, e.g. `qq.pch=1`
`qqline`	modify the `qqline` in the QQ plot, e.g. `qqline.col=0`
`label.`	modify the point labels, e.g. `label.cex=.9` or `label.font=2`
`cook.`	modify the Cook's Distance annotations. This affects only the leverage plot (`versus=3`) for `lm` models with `sta`
`caption.`	modify the overall caption (see the `caption` argument) e.g. `caption.col=2`.
`par.`	arguments for `par` (only necessary if a `par` argument name clashes with a `plotres` argument)

The `cex` argument is relative, so specifying `cex=1` is the same as not specifying `cex`.

For backwards compatibility, some dot arguments are supported but not explicitly documented.

Value

If the `which=1` plot was plotted, the return value of that plot (model dependent).

Else if the `which=3` plot was plotted, return `list(x,y)` where `x` and `y` are the coordinates of the points in that plot (but without jittering even if the `jitter` argument was used).

Else return `NULL`.

Note

This function is designed primarily for displaying standard `response - fitted` residuals for models with a single continuous response, although it will work for a few other models.

In general this function won't work on models that don't save the call and data with the model in a standard way. It uses the same underlying mechanism to access the model data as `plotmo`. For further discussion please see "*Accessing the model data*" in the plotmo vignette (also available here). Package authors may want to look at Guidelines for S3 Regression Models (also available here).

See Also

Please see the plotres vignette (also available here).

plot.lm

plot.earth

Examples

```
# we use lm in this example, but plotres is more useful for models
# that don't have a function like plot.lm for plotting residuals

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

plotres(lm.model)
```

---

plot_gbm *Plot a gbm model*

Description

Plot a gbm model showing the training and other error curves.

Usage

```
plot_gbm(object=stop("no 'object' argument"),
    smooth = c(0, 0, 0, 1),
    col = c(1, 2, 3, 4), ylim = "auto",
    legend.x = NULL, legend.y = NULL, legend.cex = .8,
    grid.col = NA,
    n.trees = NA, col.n.trees ="darkgray",
    ...)
```

Arguments

object	The gbm model.
smooth	Four-element vector specifying if smoothing should be applied to the train, test, CV, and OOB curves respectively. When smoothing is specified, a smoothed curve is plotted and the minimum is calculated from the smoothed curve. The default is c(0, 0, 0, 1) meaning apply smoothing only to the OOB curve (same as gbm.perf). Note that smooth=1 (which gets recyled to c(1,1,1,1)) will smooth all the curves.
col	Four-element vector specifying the colors for the train, test, CV, and OOB curves respectively. The default is c(1, 2, 3, 4). Use a color of 0 to remove the corresponding curve, e.g. col=c(1,2,3,0) to not display the OOB curve.

	If `col=0` (which gets recycled to `c(0,0,0,0)`) nothing will be plotted, but `plot_gbm` will return the number-of-trees at the minima as usual (as described in the Value section below).
`ylim`	The default `ylim="auto"` shows more detail around the minima. Use `ylim=NULL` for the full vertical range of the curves. Else specify `ylim` as usual.
`legend.x`	The x position of the legend. The default positions the legend automatically. Use `legend.x=NA` for no legend. See the x and y arguments of `xy.coords` for other options, for example `legend.x="topright"`.
`legend.y`	The y position of the legend.
`legend.cex`	The legend `cex` (the default is `0.8`).
`grid.col`	Default `NA`. Color of the optional grid, for example `grid.col=1`.
`n.trees`	For use by `plotres`. The x position of the gray vertical line indicating the `n.trees` passed by `plotres` to `predict.gbm` to calculate the residuals. Plotres defaults to all trees.
`col.n.trees`	For use by `plotres`. Color of the vertical line showing the `n.trees` argument. Default is `"darkgray"`.
`...`	Dot arguments are passed internally to `plot.default`.

Value

This function returns a four-element vector specifying the number of trees at the train, test, CV, and OOB minima respectively.

The minima are calculated after smoothing as specified by this function's `smooth` argument. By default, only the OOB curve is smoothed. The smoothing algorithm for the OOB curve differs slightly from `gbm.perf`, so can give a slightly different number of trees.

Note

The OOB curve

The OOB curve is artificially rescaled to force it into the plot. See Chapter 7 in the plotres vignette.

Interaction with `plotres`

When invoking this function via `plotres`, prefix any argument of `plotres` with `w1.` to tell `plotres` to pass the argument to this function. For example give `w1.ylim=c(0,10)` to `plotres` (plain `ylim=c(0,10)` in this context gets passed to the residual plots).

Acknowledgments

This function is derived from code in the `gbm` package authored by Greg Ridgeway and others.

See Also

Chapter 7 in plotres vignette discusses this function.

Examples

```
if (require(gbm)) {
    n <- 100                            # toy model for quick demo
    x1 <- 3 * runif(n)
    x2 <- 3 * runif(n)
    x3 <- sample(1:4, n, replace=TRUE)
    y <- x1 + x2 + x3 + rnorm(n, 0, .3)
    data <- data.frame(y=y, x1=x1, x2=x2, x3=x3)
    mod <- gbm(y~., data=data, distribution="gaussian",
               n.trees=300, shrinkage=.1, interaction.depth=3,
               train.fraction=.8, verbose=FALSE)

    plot_gbm(mod)

    # plotres(mod)                      # plot residuals

    # plotmo(mod)                       # plot regression surfaces
}
```

`plot_glmnet` *Plot a glmnet model*

Description

Plot the coefficient paths of a `glmnet` model.

An enhanced version of `plot.glmnet`.

Usage

```
plot_glmnet(x = stop("no 'x' argument"),
            xvar = c("rlambda", "lambda", "norm", "dev"),
            label = 10, nresponse = NA, grid.col = NA, s = NA, ...)
```

Arguments

`x`	The `glmnet` model.
`xvar`	What gets plotted along the x axis. One of: `"rlambda"` (default) decreasing log lambda (lambda is the glmnet penalty) `"lambda"` log lambda `"norm"` L1-norm of the coefficients `"dev"` percent deviance explained The default `xvar` differs from `plot.glmnet` to allow `s` to be plotted when this function is invoked by `plotres`.
`label`	Default `10`. Number of variable names displayed on the right of the plot. One of: `FALSE` display no variables

	TRUE display all variables integer (default) number of variables to display (default is 10)
nresponse	Which response to plot for multiple response models.
grid.col	Default NA. Color of the optional grid, for example grid.col="lightgray".
s	For use by plotres. The x position of the gray vertical line indicating the lambda s passed by plotres to predict.glmnet to calculate the residuals. Plotres defaults to s=0.
...	Dot arguments are passed internally to matplot. Use col to change the color of curves; for example col=1:4. The six default colors are intended to be distinguishable yet harmonious (to my eye at least), with adjacent colors as different as easily possible.

Note

Limitations

For multiple response models use the nresponse argument to specify which response should be plotted. (Currently each response must be plotted one by one.)

The type.coef argument of plot.glmnet is currently not supported.

Currently xvar="norm" is not supported for multiple response models (you will get an error message).

Interaction with plotres

When invoking this function via plotres, prefix any argument of plotres with w1. to tell plotres to pass the argument to this function. For example give w1.col=1:4 to plotres (plain col=1:4 in this context gets passed to the residual plots).

Acknowledgments

This function is based on plot.glmnet in the glmnet package authored by Jerome Friedman, Trevor Hastie, and Rob Tibshirani.

This function incorporates the function spread.labs from the orphaned package TeachingDemos written by Greg Snow.

See Also

Chapter 6 in plotres vignette discusses this function.

Examples

```
if (require(glmnet)) {
    x <- matrix(rnorm(100 * 10), 100, 10)   # n=100 p=10
    y <- x[,1] + x[,2] + 2 * rnorm(100)     # y depends only on x[,1] and x[,2]
    mod <- glmnet(x, y)

    plot_glmnet(mod)

    # plotres(mod)                          # plot the residuals
}
```

Index