Package 'nlshelper'

October 13, 2022

Title Convenient Functions for Non-Linear Regression

Version 0.2

Description A few utilities for summarizing, testing, and plotting non-linear regression models fit with nls(), nlsList() or nlme().

Depends nlme, broom, dplyr, mgcv, magicaxis

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 5.0.1

NeedsCompilation no

Author Remko Duursma [aut, cre]

Maintainer Remko Duursma <remkoduursma@gmail.com>

Repository CRAN

Date/Publication 2017-04-03 20:19:13 UTC

R topics documented:

abline_range *Add a line to a plot*

Description

As abline, but with from and to arguments. If a fitted linear regression model is used as asn argument, it uses the min and max values of the data used to fit the model.

Usage

```
abline_range(a = NULL, b = NULL, reg = NULL, from = NULL, to = NULL,
  ...)
```

Arguments

a	Intercept (optional)
b	Slope (optional)
reg	A fitted linear regression model (output of lm).
from	Draw from this X value
to	Draw to this x value
...	Further parameters passed to segments

See Also

See add_regres_line for adding a regression line with a confidence interval

Examples

```
# Add a line manually
with(mtcars, plot(1/wt, mpg, xlim=c(0,0.8), ylim=c(0,40)))
abline_range(0,50,from=0.2, to=0.6)

# Add a line across the range of the data from a regression object
with(mtcars, plot(1/wt, mpg, xlim=c(0,0.8), ylim=c(0,40)))
fit <- lm(mpg ~ I(1/wt), data=mtcars)
abline_range(fit)
```

add_regres_line *Add a regression line and confidence band to a plot*

Description

Plots a regression line from a simple linear model (of the form `lm(y ~ x)`) to a plot. Also plots the confidence band for the mean, which is calculated using `predict.lm`.

Usage

```
add_regres_line(fit, from = NULL, to = NULL, band = TRUE,
  ci.col = "#BEBEBEB3", ...)
```

Arguments

`fit`	Object returned by lm. Only models of the form `y ~ x` are supported, without expressions in `I()` (see Examples), or interactions, or multiple variables.
`from`	Optional (read from fitted model); Draw from this X value.
`to`	Optional (read from fitted model); Draw to this x value.
`band`	Logical. Whether to add a confidence band.
`ci.col`	Colour of the confidence band, if plotted. Defaults to a transparent grey colour.
`...`	Further arguments passed to `abline_range`

Examples

```
#'Add a line across the range of the data from a regression object
with(mtcars, plot(1/wt, mpg, xlim=c(0,0.8), ylim=c(0,40)))

# add_regres_line does not allow I() expressions; yet.
mtcars$inv_wt <- 1 / mtcars$wt
fit <- lm(mpg ~ inv_wt, data=mtcars)
add_regres_line(fit)

# Add the regression line and confidence band behind the data
fit <- lm(height ~ age, data=Loblolly)
with(Loblolly, plot(age, height, pch=19, panel.first=add_regres_line(fit)))
```

anova_nlslist *Anova for nlsList*

Description

Applies an F-test to a non-linear regression model that includes a grouping variable (fit with `nlsList`), comparing it to a model without a grouping variable. This is a convenient way to test whether there is an overall effect of the grouping variable on the non-linear relationship.

Usage

```
anova_nlslist(nlsfull, nlsreduc)
```

Arguments

`nlsfull`	The full model, an object returned by `nlsList`
`nlsreduc`	The reduced model, which is identical to the full model except the grouping variable has been removed, and it was fit with `nls`

Examples

```
chick <- as.data.frame(ChickWeight)

# Fit a simple model with nls
fit0 <- nls(weight ~ a*Time^b, data=chick, start=list(a=10, b=1.1))

# Fit an nlsList model, with a grouping variable (Diet)
fit1 <- nlsList(weight ~ a*Time^b | Diet, data=chick, start=list(a=10, b=1.1))

# Using an F-test, test whether the fit is significantly better when adding
# a grouping variable
anova_nlslist(fit1, fit0)
```

plot_gam *Plot a generalized additive model*

Description

This is a simple wrapper to fit and plot a basic type of generalized additive model. The fitted model is of the form `gam(Y ~ s(X, k))`, which can be fitted by a specified grouping variable (using the g argument). Also supported is an optional random effect, in which case the model fitted is `gamm(Y ~ s(X, k=k), random = list(R=~1), data=dfr)`.

Usage

```
plot_gam(x, y, g = NULL, data, fittype = c("gam", "lm"), kgam = 4,
  R = NULL, log = "", axes = TRUE, fitoneline = FALSE,
  points.col = NULL, lines.col = NULL, ci.col = "#D3D3D3B3",
  xlab = NULL, ylab = NULL, band = TRUE, plotit = TRUE, add = FALSE,
  npred = 101, lwd = 2, ...)
```

Arguments

x	Variable for X axis (unquoted)
y	Variable for Y axis (unquoted)
g	Variable for grouping (unquoted); optional
data	Dataframe containing x and y
fittype	Either 'gam' (default), or 'lm' in which case a simple linear model is fit - useful for comparison.
kgam	the k parameter for smooth terms in gam.
R	An optional random effect (quoted)
log	Whether to add log axes for x or y (but no transformations are done).
axes	Logical (default TRUE), whether to add axes to the plot.
fitoneline	Whether to fit only one curve to the entire dataset, regardless of whether a grouping variable was defined. Default FALSE.
points.col	Colours of the points, can be a vector (one value for each group, if present).
lines.col	Colours of the lines, can be a vector (one value for each group, if present).
ci.col	Colour of the confidence band, if plotted. Defaults to a transparent grey colour.
xlab	X-axis label
ylab	Y-axis label
band	Logical. If true, plots the confidence band (as a transparent polygon).
plotit	Logical (default TRUE); if FALSE, suppresses the plot.
add	Logical (default FALSE), if TRUE, adds to an existing plot.
npred	Number of x values to use for prediction
lwd	Line thickness (see par)
...	Further arguments passed to plot or points, for example to change colour of plotting symbols.

Details

In either case, the k parameter necessary for the GAM fit can be set using the kgam argument. Seechoose.k for details on this setting (it is important you don't just use the default value here!).

Examples

```
data(Loblolly)
plot_gam(age, height, data=Loblolly)
plot_gam(age, height, Seed, data=Loblolly, band=FALSE, lines.col="black")
plot_gam(age, height, Seed, data=Loblolly, band=FALSE, lines.col="black", fittype="lm")

data(ChickWeight)
plot_gam(Time, weight, Diet, R="Chick", data=ChickWeight, lines.col=rainbow(4))
```

`plot_nls` *Plot a non-linear or non-parametric regression model*

Description

Convenient function for adding curves to an existing plot, or to plot the data with the fitted curve. For non-linear regression plotting (`plot_nls`), works for simple non-linear regression models fit with `nls`, and grouped non-linear regression (with `nlsList`), in which case one fitted curve for each group is drawn on the same plot. For local regression models fitted with `loess`, use the `plot_loess` function which additionally adds a confidence interval around the fitted curve.

Usage

```
plot_nls(object, col = NULL, band = TRUE, plotdata = TRUE,
  lines.col = palette(), points.col = palette(), ci.col = "#BEBEBEB3",
  lwd = 1, lty = 1, add = FALSE, xlab = NULL, ylab = NULL,
  coverage = 0.95, ...)

plot_loess(object, ...)
```

Arguments

`object`	The object returned by `nls`, `nlsList` or `loess`
`col`	Colour to be used for the data symbols and the fitted line, unless `lines.col` and `points.col` are provided
`band`	For `plot_loess`, whether to add a confidence band. Not yet implemented for `plot_nls`
`plotdata`	Logical. Whether to add the data points to the plot.
`lines.col`	Colour(s) for the fitted lines. When plotting a `nlsList` object, can be a vector that represents colours for each group.
`points.col`	Colour(s) for the data symbols. When plotting a `nlsList` object, can be a vector that represents colours for each group.
`ci.col`	Colour of the confidence band, if plotted. Defaults to a transparent grey colour.
`lwd`	Thickness of the line (see `par`)
`lty`	Line type (see `par`)

add	Logical. Whether to add to current plot (default FALSE).
xlab	Label for x-axis
ylab	Label for y-axis
coverage	If confidence band to be plotted, the coverage (e.g. for 95% confidence interval, use 0.95)
...	Further arguments passed to plot

Value

Returns the predicted values used in plotting (invisibly), as a dataframe with columns 'predvar' (regularly spaced predictor values), and 'fit' (fitted values). For plot_loess also returns confidence intervals, standard error, and df of the residual.

Examples

```
# Plot an nls object
chick <- as.data.frame(ChickWeight)
fit0 <- nls(weight ~ a*Time^b, data=chick, start=list(a=10, b=1.1))
plot_nls(fit0)

# Plot a grouped nls object
library(nlme)
fit1 <- nlsList(weight ~ a*Time^b|Diet, data=chick, start=list(a=10, b=1.1))
plot_nls(fit1)

# Plot a local regression object, with confidence interval
l <- loess(wt ~ disp, data=mtcars)
plot_loess(l)

# To plot behind the data:
with(mtcars, plot(disp, wt, pch=19,
 panel.first=plot_loess(l, plotdata=FALSE)))
```

---

tidy.nlsList *Tidy method for nlsList*

Description

Adds a method to tidy (broom package), so that we can use it for models fitted with nlsList.

Usage

```
## S3 method for class 'nlsList'
tidy(x, conf.int = FALSE, conf.level = 0.95,
  quick = FALSE, ...)
```

Arguments

x	An object returned by nlsList
conf.int	Whether to calculate confidence intervals
conf.level	The level of the confidence interval
quick	If TRUE, only returns the coefficients.
...	Further arguments passed to tidy

Examples

```
chick <- as.data.frame(ChickWeight)

# Fit an nlsList model, with a grouping variable (Diet)
fit1 <- nlsList(weight ~ a*Time^b | Diet, data=chick, start=list(a=10, b=1.1))

# Collect coefficients
tidy(fit1)

# ... and confidence intervals
tidy(fit1, conf.int=TRUE)
```

Index