Package 'investr'

September 15, 2014

Type Package

Title Inverse Estimation/Calibration Functions

Version 1.1.2

Author Brandon M. Greenwell

Maintainer Brandon M. Greenwell <greenwell.brandon@gmail.com>

Description Functions to facilitate inverse estimation/calibration in linear,nonlinear, and (linear) mixed models. A generic function is also provided
for plotting fitted regression models with or without confidence/prediction
bands that may be of use to the general user.

Depends graphics, stats

Suggests boot, car, testthat

Imports nlme

Date 2014-04-18

License GPL (>= 2)

URL https://github.com/w108bmg/investr

LazyLoad true

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-15 07:26:12

R topics documented:

`investr-package` *Inverse estimation/calibration functions*

Description

This package provides functions to facilitate inverse estimation/calibration with linear, nonlinear, and (linear) mixed models.

Details

Package:	investr
Type:	Package
Version:	1.1.1
Date:	2014-04-18
License:	GPL (>= 2)

Author(s)

Brandon M. Greenwell

Maintainer: Brandon M. Greenwell <brandon.greenwell@afit.edu>

References

Bates, D. M., and Watts, D. G. Nonlinear Regression Analysis and its Applications. New York: Wiley, 2007.

Baty, F. and Delignette-Muller, M. L. (2012), `nlstools`: tools for nonlinear regression.

Graybill, F. A., and Iyer, H. K. Regression Analysis: Concepts and Applications. Belmont, Calif: Duxbury Press, 1994.

Huet, S., Bouvier, A., Poursat, M-A., and Jolivet, E. Statistical Tools for Nonlinear Regression: A Practical Guide with S-PLUS and R Examples. New York: Springer, 2004.

Seber, G. A. F., and Wild, C. J. Nonlinear regression. New York: Wiley, 1989.

arsenic *Concentrations of arsenic in water samples*

Description

The data give the actual and measured concentrations of arsenic present in water samples.

Format

A data frame with 32 rows and 2 variables

Details

- `actual` True amount of arsenic present
- `measured` Measured amount of arsenic present

References

Graybill, F. A., and Iyer, H. K. (1994) *Regression analysis: Concepts and Applications*. Duxbury Press.

calibrate *Calibration for the simple linear regression model.*

Description

The function `calibrate` computes the maximum likelihood estimate and a condfidence interval for the unknown predictor value that corresponds to an observed value of the response (or vector thereof) or specified value of the mean response. See the reference listed below for more details.

Usage

```
calibrate(object, ...)

## Default S3 method:
calibrate(object, y0, interval = c("inversion", "Wald",
  "none"), level = 0.95, mean.response = FALSE, adjust = c("none",
  "Bonferroni", "Scheffe"), k, ...)

## S3 method for class 'formula'
calibrate(formula, data = NULL, ..., subset,
  na.action = na.fail)

## S3 method for class 'lm'
calibrate(object, ...)
```

Arguments

object	An object that inherits from class lm, a matrix, a list, or a data frame.
...	Additional optional arguments. At present, no optional arguments are used.
y0	The value of the observed response(s) or specified value of the mean response.
interval	The method to use for forming a confidence interval.
level	A numeric scalar between 0 and 1 giving the confidence level for the interval to be calculated.
mean.response	Logicial indicating whether confidence intervals should correspond to an observed response(s) (FALSE) or a specified value of the mean response (TRUE). Default is FALSE.
adjust	A logical value indicating if an adjustment should be made to the critical value used in calculating the confidence interval. This useful for when the calibration curve is to be used multiple, say k, times.
k	The number times the calibration curve is to be used for computing a confidence interval. Only needed when adjust = TRUE.
formula	A formula of the form y ~ x.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which lm is called.
subset	An optional vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain NAs.

Value

An object of class calibrate containing the following components:

- estimate The estimate of x0.
- lwr The lower confidence limit for x0.
- upr The upper confidence limit for x0.
- se An estimate of the standard error (Wald interval only).
- interval The method used for calculating lower and upper (only used by print method).

Warning

You must not call this function unless ...

Note

The function invest is more general, but based on numerical techniques to find the solution. When the underlying model is that of the simple linear regression model with normal errors, closed-form expressions exist which are utilized by the function calibrate.

References

Graybill, F. A., and Iyer, H. K. (1994) *Regression analysis: Concepts and Applications*. Duxbury Press.

Miller, R. G. (1981) *Simultaneous Statistical Inference*. Springer-Verlag.

Examples

```
## Inverting a prediction interval for an individual response
arsenic.lm <- lm(measured ~ actual, data = arsenic)
plotFit(arsenic.lm, interval = "prediction", shade = TRUE,
        col.pred = "lightblue")
calibrate(arsenic.lm, y0 = 3, interval = "inversion")

## Inverting a confidence interval for the mean response
crystal.lm <- lm(weight ~ time, data = crystal)
plotFit(crystal.lm, interval = "confidence", shade = TRUE,
        col.conf = "lightblue")
calibrate(crystal.lm, y0 = 8, interval = "inversion", mean.response = TRUE)

## Wald interval and approximate standard error based on the delta method
calibrate(crystal.lm, y0 = 8, interval = "Wald", mean.response = TRUE)

## Alterntively, we can use the car package to compute the standard error (this
## is trickier though when mean.respone = FALSE, hence, it is better to use the
## calibrate function).
library(car)
deltaMethod(crystal.lm, g = "(8 - b0) / b1", parameterNames = c("b0", "b1"))
```

`crystal` *Crystal weight data*

Description

The data give the growing time and final weight of crystals.

Format

A data frame with 14 rows and 2 variables

Details

- `time` Time taken to grow (hours).
- `weight` Final weight of the crystal (grams).

References

Graybill, F. A., and Iyer, H. K. (1994) *Regression analysis: Concepts and Applications*. Duxbury Press.

`invest` *Calibration for Linear and Nonlinear Regression Models.*

Description

The function `invest` computes the inverse estimate and a condfidence interval for the unknown predictor value that corresponds to an observed value of the response (or vector thereof) or specified value of the mean response. See the references listed below for more details.

Usage

```
invest(object, ...)

## S3 method for class 'lm'
invest(object, y0, interval = c("inversion", "Wald", "none"),
  level = 0.95, mean.response = FALSE, data, lower, upper,
  tol = .Machine$double.eps^0.25, maxiter = 1000, adjust = c("none",
  "Bonferroni"), k, ...)

## S3 method for class 'nls'
invest(object, y0, interval = c("inversion", "Wald", "none"),
  level = 0.95, mean.response = FALSE, data, lower, upper,
  tol = .Machine$double.eps^0.25, maxiter = 1000, adjust = c("none",
  "Bonferroni"), k, ...)

## S3 method for class 'lme'
invest(object, y0, interval = c("inversion", "Wald", "none"),
  level = 0.95, mean.response = FALSE, data, lower, upper, q1, q2,
  tol = .Machine$double.eps^0.25, maxiter = 1000, ...)
```

Arguments

`object`	An object that inherits from class `lm` or `nls`.
`...`	Additional optional arguments. At present, no optional arguments are used.
`y0`	The value of the observed response(s) or specified value of the mean response.
`interval`	The type of interval required.
`level`	A numeric scalar between 0 and 1 giving the confidence level for the interval to be calculated.
`mean.response`	Logical indicating whether confidence intervals should correspond to an individual response (`FALSE`) or a mean response (`TRUE`).
`data`	An optional data frame. This is required if `object$data` is `NULL`.

lower	The lower endpoint of the interval to be searched.
upper	The upper endpoint of the interval to be searched.
tol	The desired accuracy passed on to uniroot. Recommend a minimum of 1e-10.
maxiter	The maximum number of iterations passed on to uniroot.
adjust	A logical value indicating if an adjustment should be made to the critical value used in calculating the confidence interval. This is useful for when the calibration curve is to be used multiple, say k, times.
k	The number times the calibration curve is to be used for computing a confidence interval. Only needed when adjust = "Bonferroni".
q1	Optional lower cutoff to be used in forming confidence intervals. Only used when object inherits from class lme. Defaults to qnorm((1+level)/2).
q2	Optional upper cutoff to be used in forming confidence intervals. Only used when object inherits from class lme. Defaults to qnorm((1-level)/2).

Value

An object of class calibrate containing the following components:

- estimate The estimate of x0.
- lwr The lower confidence limit for x0.
- upr The upper confidence limit for x0.
- se An estimate of the standard error (Wald interval only).
- interval The method used for calculating lower and upper (only used by print method).

References

Graybill, F. A., and Iyer, H. K. (1994) *Regression analysis: Concepts and Applications*. Duxbury Press.

Huet, S., Bouvier, A., Poursat, M-A., and Jolivet, E. (2004) *Statistical Tools for Nonlinear Regression: A Practical Guide with S-PLUS and R Examples*. Springer.

Seber, G. A. F., and Wild, C. J. (1989) *Nonlinear regression*. Wiley.

Oman, Samuel D. (1998). Calibration with Random Slopes. *Biometrics* **85**(2): 439–449. doi:10.1093/biomet/85.2.439.

Examples

```
data(Puromycin, package = "datasets")
Puromycin2 <- Puromycin[Puromycin$state == "treated", ]
Puro2.nls <- nls(rate ~ (theta1 * conc) / (theta2 + conc),
                 data = Puromycin2, start = list(theta1 = 200, theta2 = 1))
plotFit(Puro2.nls, interval = "both")
invest(Puro2.nls, y0 = 100, interval = "inversion")
invest(Puro2.nls, y0 = 100, interval = "inversion", mean.response = TRUE)
```

plotFit *Plotting Confidence/Prediction Bands*

Description

Plots fitted model for an object of class `lm` or `nls` with the option of adding a confidence and/or prediction band.

Usage

```
plotFit(object, ...)

## S3 method for class 'lm'
plotFit(object, interval = c("none", "both", "confidence",
  "prediction"), level = 0.95, data, adjust = c("none", "Bonferroni",
  "Scheffe"), k, ..., shade = FALSE, extend.range = FALSE, hide = TRUE,
  col.conf = if (shade) grey(0.7) else "black", col.pred = if (shade)
  grey(0.9) else "black", border.conf = col.conf, border.pred = col.pred,
  col.fit = "black", lty.conf = if (shade) 1 else 2, lty.pred = if (shade)
  1 else 3, lty.fit = 1, lwd.conf = 1, lwd.pred = 1, lwd.fit = 1,
  n = 500, xlab, ylab, xlim, ylim)

## S3 method for class 'nls'
plotFit(object, interval = c("none", "both", "confidence",
  "prediction"), level = 0.95, data, adjust = c("none", "Bonferroni",
  "Scheffe"), k, ..., shade = FALSE, extend.range = FALSE, hide = TRUE,
  col.conf = if (shade) grey(0.7) else "black", col.pred = if (shade)
  grey(0.9) else "black", border.conf = col.conf, border.pred = col.pred,
  col.fit = "black", lty.conf = if (shade) 1 else 2, lty.pred = if (shade)
  1 else 3, lty.fit = 1, lwd.conf = 1, lwd.pred = 1, lwd.fit = 1,
  n = 500, xlab, ylab, xlim, ylim)
```

Arguments

`object`	An object that inherits from class `lm` or `nls`.
`...`	Additional optional arguments passed on to `plot`.
`interval`	A character string indicating if a prediction band, confidence band, both, or none should be plotted.
`level`	The desired confidence level.
`data`	An optional data frame containing the variables in the model.
`adjust`	A character string indicating the type of adjustment (if any) to make to the confidence/prediction bands.
`k`	An integer to be used in computing the critical value for the confidence/prediction bands. Only needed when `adjust = "Bonferroni"` or when `adjust = "Scheffe"` and `interval = "prediction"`.

shade	A logical value indicating if the band should be shaded.
extend.range	A logical value indicating if the fitted regression line and bands (if any) should extend to the edges of the plot. Default is FALSE.
hide	A logical value indicating if the fitted model should be plotted on top of the points (FALSE) or behind them (TRUE). Default is TRUE.
col.conf	Shade color for confidence band.
col.pred	Shade color for prediction band.
border.conf	The color to use for the confidence band border.
border.pred	The color to use for the prediction band border.
col.fit	The color to use for the fitted line.
lty.conf	Line type to use for confidence band border.
lty.pred	Line type to use for prediction band border.
lty.fit	Line type to use for the fitted regression line.
lwd.conf	Line width to use for confidence band border.
lwd.pred	Line width to use for prediction band border.
lwd.fit	Line width to use for the fitted regression line.
n	The number of predictor values at which to evaluate the fitted model (larger implies a smoother plot).
xlab	A title for the x axis.
ylab	A title for the y axis.
xlim	The x limits (x1, x2) of the plot.
ylim	The y limits (y1, y2) of the plot.

Note

By default, the plotted intervals are pointwise intervals. For simultaneous intervals use `adjust = "Bonferroni"` or `adjust = "Scheffe"`. For the Bonferroni adjustment, you must specify a value for `k`, the number of intervals for which the coverage is to hold simultaneously. For the Scheffe adjustment, specifying a value for `k` is only required when `interval = "prediction"`; if `interval = "confidence"`, `k` is set equal to p, the number of regression parameters. For example, if `object` is a simple linear regression model, then calling `plotFit` with `interval = "confidence"` and `adjust = "Scheffe"` will plot the Working-Hotelling band.

Confidence/prediction bands for nonlinear regression (i.e., objects of class `nls`) are based on a linear approximation as described in Bates & Watts (2007). This funtion was inpired by the `plotfit` function from the `nlstools` package.

References

Bates, D. M., and Watts, D. G. (2007) *Nonlinear Regression Analysis and its Applications*. Wiley.

F. Baty and M. L. Delignette-Muller (2012), A Toolbox for Nonlinear Regression in R: The Package nlstools. *Journal of Statistical Software* **(under revision)**.

Examples

```
## A linear regression example
data(cars, package = "datasets")
library(splines)
cars.lm1 <- lm(dist ~ speed, data = cars)
cars.lm2 <- lm(dist ~ speed + I(speed^2), data = cars)
cars.lm3 <- lm(dist ~ speed + I(speed^2) + I(speed^3), data = cars)
cars.lm4 <- lm(dist ~ ns(speed, df = 3), data = cars)
par(mfrow = c(2, 2))
plotFit(cars.lm1, interval = "both", xlim = c(-10, 40), ylim = c(-50, 150),
        main = "linear")
plotFit(cars.lm2, interval = "both", xlim = c(-10, 40), ylim = c(-50, 150),
        main = "quadratic")
plotFit(cars.lm3, interval = "both", xlim = c(-10, 40), ylim = c(-50, 150),
        main = "cubic")
plotFit(cars.lm4, interval = "both", xlim = c(-10, 40), ylim = c(-50, 150),
        main = "cubic spline")

## A nonlinear regression example
par(mfrow = c(1, 1))
library(RColorBrewer) # requires that RColorBrewer be installed
blues <- brewer.pal(9, "Blues")
data(Puromycin, package = "datasets")
Puromycin2 <- Puromycin[Puromycin$state == "treated", ][, 1:2]
Puro.nls <- nls(rate ~ Vm * conc/(K + conc), data = Puromycin2,
                start = c(Vm = 200, K = 0.05))
plotFit(Puro.nls, interval = "both", pch = 19, shade = TRUE,
        col.conf = blues[4], col.pred = blues[2])
```

Index