Package 'investr'

October 13, 2022

Type Package

Title Inverse Estimation/Calibration Functions

Version 1.4.2

Author Brandon M. Greenwell

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

Description Functions to facilitate inverse estimation (e.g., calibration) in linear, generalized linear, nonlinear, and (linear) mixed-effects 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. For a general overview of these methods, see Greenwell and Schubert Kabban (2014) <doi:10.32614/RJ-2014-009>.

License GPL (>= 2)

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

Depends R (>= 3.3.0)

Suggests boot, datasets, knitr, MASS, testthat

Imports graphics, nlme, stats, utils

LazyLoad true

LazyData true

RoxygenNote 7.1.2

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2022-03-31 07:30:10 UTC

R topics documented:

`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 columns.

Details

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

Source

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

`beetle` *Dobson's Beetle Data*

Description

The data give the number of flour beetles killed after five hour exposure to the insecticide carbon disulphide at eight different concentrations.

Format

A data frame with 8 rows and 3 columns.

Details

- `ldose` Log dose of carbon disulphide.
- `y` Number of beetles subjected to insecticide.
- `n` Number of beetles killed.

Source

Dobson, A. (2002) *An Introduction to Generalized Linear Models*. Chapman & Hall/CRC.

`bladder` *Bladder volume data*

Description

A series of 23 women patients attending a urodynamic clinic were recruited for a study. After successful voiding of the bladder, sterile water was introduced in additions of 10, 15, and then 25 ml increments up to a final cumulative total of 175 ml. At each volume a measure of height (H) in mm and depth (D) in mm of largest ultrasound bladder images were taken. The product H × D was taken as a measure of liquid volume.

Format

A data frame with 184 rows and 3 columns.

Details

- `subject` The subject ID.
- `HD` The product H × D (mm^2).
- `volume` The true volume of sterile water in the bladder (ml).

Source

Brown, P. (1993) *Measurement, Regression, and Calibration*. Oxford University 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 = stats::na.fail)

## S3 method for class 'lm'
calibrate(
  object,
  y0,
  interval = c("inversion", "Wald", "none"),
  level = 0.95,
  mean.response = FALSE,
  adjust = c("none", "Bonferroni", "Scheffe"),
  k,
  ...
)
```

Arguments

`object`	A matrix, list, data frame, or object that inherits from class `lm`.
`...`	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 constructing the confidence interval. This useful when the calibration curve is to be used k > 0 times.
k	The number of 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 was 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 "invest" 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).

Note

The invest function is more general, but is 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 calibrate function.

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

```
#
# Arsenic example (simple linear regression with replication)
#

# Inverting a prediction interval for an individual response
arsenic.lm <- stats::lm(measured ~ actual, data = arsenic)
plotFit(arsenic.lm, interval = "prediction", shade = TRUE,
        col.pred = "lightblue")
(cal <- calibrate(arsenic.lm, y0 = 3, interval = "inversion"))
abline(h = 3)
segments(cal$estimate, 3, cal$estimate, par()$usr[3])
arrows(cal$lower, 3, cal$lower, par()$usr[3])
arrows(cal$upper, 3, cal$upper, par()$usr[3])

#
# Crystal weight example (simple linear regression)
#

# Inverting a confidence interval for the mean response
crystal.lm <- stats::lm(weight ~ time, data = crystal)
plotFit(crystal.lm, interval = "confidence", shade = TRUE,
        col.conf = "lightblue")
(cal <- calibrate(crystal.lm, y0 = 8, interval = "inversion",
                  mean.response = TRUE))
abline(h = 8)
segments(cal$estimate, 8, cal$estimate, par()$usr[3])
arrows(cal$lower, 8, cal$lower, par()$usr[3])
arrows(cal$upper, 8, cal$upper, par()$usr[3])

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

`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 columns.

Details

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

Source

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

`invest` *Inverse estimation for linear, nonlinear, and generalized linear models*

Description

Provides point and interval estimates 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, y0, ...)

## S3 method for class 'lm'
invest(
  object,
  y0,
  interval = c("inversion", "Wald", "percentile", "none"),
  level = 0.95,
  mean.response = FALSE,
  x0.name,
  newdata,
  data,
  boot.type = c("parametric", "nonparametric"),
  nsim = 999,
  seed = NULL,
  progress = FALSE,
  lower,
  upper,
  extendInt = "no",
  tol = .Machine$double.eps^0.25,
  maxiter = 1000,
  adjust = c("none", "Bonferroni"),
  k,
  ...
)

## S3 method for class 'glm'
invest(
  object,
  y0,
  interval = c("inversion", "Wald", "percentile", "none"),
```

```
  level = 0.95,
  lower,
  upper,
  x0.name,
  newdata,
  data,
  extendInt = "no",
  tol = .Machine$double.eps^0.25,
  maxiter = 1000,
  ...
)

## S3 method for class 'nls'
invest(
  object,
  y0,
  interval = c("inversion", "Wald", "percentile", "none"),
  level = 0.95,
  mean.response = FALSE,
  data,
  boot.type = c("parametric", "nonparametric"),
  nsim = 1,
  seed = NULL,
  progress = FALSE,
  lower,
  upper,
  extendInt = "no",
  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", "percentile", "none"),
  level = 0.95,
  mean.response = FALSE,
  data,
  lower,
  upper,
  q1,
  q2,
  extendInt = "no",
  tol = .Machine$double.eps^0.25,
```

```
  maxiter = 1000,
  ...
)
```

Arguments

object	An object that inherits from class lm, glm, nls, or lme.
y0	The value of the observed response(s) or specified value of the mean response. For glm objects, y0 should be on the scale of the response variable (e.g., a number between 0 and 1 for binomial families).
...	Additional optional arguments. At present, no optional arguments are used.
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). For glm objects, this is always TRUE.
x0.name	For multiple linear regression, a character string giving the name of the predictor variable of interest.
newdata	For multiple linear regression, a data.frame giving the values of interest for all other predictor variables (i.e., those other than x0.name).
data	An optional data frame. This is required if object$data is NULL.
boot.type	Character string specifying the type of bootstrap to use when interval = "percentile". Options are "parametric" and "nonparametric".
nsim	Positive integer specifying the number of bootstrap simulations; the bootstrap B (or R).
seed	Optional argument to set.seed.
progress	Logical indicating whether to display a text-based progress bar during the bootstrap simulation.
lower	The lower endpoint of the interval to be searched.
upper	The upper endpoint of the interval to be searched.
extendInt	Character string specifying if the interval c(lower, upper) should be extended or directly produce an error when the inverse of the prediction function does not have differing signs at the endpoints. The default, "no", keeps the search interval and hence produces an error. Can be abbreviated. See the documentation for the base R function uniroot for details.
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 stats::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 stats::qnorm((1-level)/2).

Value

Returns an object of class "invest" or, if interval = "percentile", of class c("invest", "bootCal"). The generic function {plot} can be used to plot the output of the bootstrap simulation when interval = "percentile".

An object of class "invest" 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 and percentile intervals only).
- bias The bootstrap estimate of bias (percentile interval only).
- bootreps Vector of bootstrap replicates (percentile interval only).
- nsim The number of bootstrap replicates (percentile interval only).
- interval The method used for calculating lower and upper (only used by {print} method).

References

Greenwell, B. M. (2014). *Topics in Statistical Calibration*. Ph.D. thesis, Air Force Institute of Technology. URL https://apps.dtic.mil/sti/pdfs/ADA598921.pdf

Greenwell, B. M., and Schubert Kabban, C. M. (2014). investr: An R Package for Inverse Estimation. *The R Journal*, **6**(1), 90–100. URL http://journal.r-project.org/archive/2014-1/greenwell-kabban.pdf.

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.

Norman, D. R., and Smith H. (2014). *Applied Regression Analysis*. John Wiley & Sons.

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

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

Examples

```
#
# Dobson's beetle data (generalized linear model)
#

# Complementary log-log model
mod <- glm(cbind(y, n-y) ~ ldose, data = beetle,
           family = binomial(link = "cloglog"))
```

```
plotFit(mod, pch = 19, cex = 1.2, lwd = 2,
        xlab = "Log dose of carbon disulphide",
        interval = "confidence", shade = TRUE,
        col.conf = "lightskyblue")

# Approximate 95% confidence intervals and standard error for LD50
invest(mod, y0 = 0.5)
invest(mod, y0 = 0.5, interval = "Wald")

#
# Nasturtium example (nonlinear least-squares with replication)
#

# Log-logistic model
mod <- nls(weight ~ theta1/(1 + exp(theta2 + theta3 * log(conc))),
           start = list(theta1 = 1000, theta2 = -1, theta3 = 1),
           data = nasturtium)
plotFit(mod, lwd.fit = 2)

# Compute approximate 95% calibration intervals
invest(mod, y0 = c(309, 296, 419), interval = "inversion")
invest(mod, y0 = c(309, 296, 419), interval = "Wald")

# Bootstrap calibration intervals. In general, nsim should be as large as
# reasonably possible (say, nsim = 9999).
boo <- invest(mod, y0 = c(309, 296, 419), interval = "percentile",
              nsim = 300, seed = 101)
boo  # print bootstrap summary
plot(boo)  # plot results

#
# Bladder volume example (random coefficient model)
#

# Load required packages
library(nlme)

# Plot data
plot(HD^(3/2) ~ volume, data = bladder, pch = 19,
       col = adjustcolor("black", alpha.f = 0.5))

# Fit a random intercept and slope model
bladder <- na.omit(bladder)
ris <- lme(HD^(3/2) ~ volume, data = bladder, random = ~volume|subject)
invest(ris, y0 = 500)
invest(ris, y0 = 500, interval = "Wald")
```

`investr` *investr: a package for inverse estimation in R*

Description

Inverse estimation, also referred to as the calibration problem, is a classical and well-known problem in regression. In simple terms, it involves the use of an observed value of the response (or specified value of the mean response) to make inference on the corresponding unknown value of the explanatory variable.

Details

A detailed introduction to investr has been published in The R Journal: "investr: An R Package for Inverse Estimation." You can track development at `https://github.com/bgreenwell/investr`. To report bugs or issues, contact the main author directly or submit them to `https://github.com/bgreenwell/investr/issues`.

As of right now, `investr` supports (univariate) inverse estimation with objects of class:

- `lm` — linear models (multiple predictor variables allowed)
- `glm` — generalized linear models (multiple predictor variables allowed)
- `nls` — nonlinear least-squares models
- `lme` — linear mixed-effects models (fit using the `nlme` package)

`nasturtium` *Bioassay on Nasturtium*

Description

The data give the actual concentrations of an agrochemical present in soil samples versus the weight of the plant after three weeks of growth.

Format

A data frame with 42 rows and 2 columns.

Details

- `conc` True concentration of agrochemical (g/ha).
- `weight` Weight of plant (mg) after 3 weeks' growth.

Source

Racine-Poon, A. (1988) A Bayesian Approach to Nonlinear Calibration Problems, *Journal of the American Statistical Association*, **83**, 650–656.

References

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.

plot.bootCal *Plots method for bootstrap calibration*

Description

The plot method for "bootCal" objects. In particular, this method takes a "bootCal" object and produces plots for the bootstrap replicates of the inverse estimate.

Usage

```
## S3 method for class 'bootCal'
plot(x, ...)
```

Arguments

x	An object that inherits from class "bootCal".
...	Additional optional arguments. At present, no optional arguments are used.

Value

x is returned invisibly.

plotFit *Plotting fitted models*

Description

Generic function for plotting predictions from various types of fitted models. plotFit currently supports objects of class lm, glm, and nls. A default method also exists which may be used for plotting the fitted mean response from other model fits (e.g., lqs and rlm from the **MASS** package).

Usage

```
plotFit(object, ...)

## Default S3 method:
plotFit(
  object,
  type = c("response", "link"),
  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) grDevices::grey(0.7) else "black",
  col.pred = if (shade) grDevices::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	A fitted model object. Typically, an object that inherits from class lm, glm, or nls (but others may work too).
...	Additional optional arguments passed on to plot.
type	The type of prediction required. The default is on the scale of the response variable; the alternative "link" is on the scale of the linear predictor. This option is only used when plotting glm objects.
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 gives 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.

Value

No return value (called for side effects).

Note

By default, the plotted intervals are unadjusted (i.e., 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 the linear approximation described in Bates & Watts (2007).

References

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

Florent Baty, Christian Ritz, Sandrine Charles, Martin Brutsche, Jean-Pierre Flandrois, Marie-Laure Delignette-Muller (2015). A Toolbox for Nonlinear Regression in R: The Package nlstools. *Journal of Statistical Software*, **66**(5), 1-21.

See Also

`plotfit`

Examples

```
# A nonlinear least squares example (see ?datasets::Puromycin and
# ?investr::predFit)
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 = "skyblue4", col.pred = "lightskyblue2")
```

---

predFit *Predictions from a Fitted Model*

Description

Generic prediction method for various types of fitted models. predFit can be used to obtain standard errors of fitted values and adjusted/unadjusted confidence/prediction intervals for objects of class "lm", "nls", and "glm".

Usage

```
predFit(object, ...)

## Default S3 method:
predFit(object, ...)

## S3 method for class 'lm'
predFit(
  object,
  newdata,
  se.fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  level = 0.95,
  adjust = c("none", "Bonferroni", "Scheffe"),
  k,
  ...
)

## S3 method for class 'glm'
predFit(
  object,
  newdata,
  type = c("link", "response"),
  se.fit = FALSE,
  interval = c("none", "confidence"),
  level = 0.95,
  ...
```

```
)

## S3 method for class 'nls'
predFit(
  object,
  newdata,
  se.fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  level = 0.95,
  adjust = c("none", "Bonferroni", "Scheffe"),
  k,
  ...
)

## S3 method for class 'lme'
predFit(object, newdata, se.fit = FALSE, ...)
```

Arguments

object	An object that inherits from class "lm", "glm", "nls", or "lme".
...	Additional optional arguments. At present, no optional arguments are used.
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
se.fit	A logical vaue indicating if standard errors are required. Default is FALSE.
interval	Type of interval to be calculated. Can be one of "none" (default), "confidence", or "prediction". Default is "none".
level	A numeric scalar between 0 and 1 giving the confidence level for the intervals (if any) to be calculated. Default is 0.95.
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. Default is FALSE.
k	The number times the calibration curve is to be used for computing a confidence/prediction interval. Only needed when adjust = "Bonferroni".
type	Character string specifying the type of prediction. Current options are type = "link" (the default) and type = "response".

Details

Confidence and prediction intervals for linear models (i.e., "lm" objects) are obtained according to the usual formulas. Nonlinear and generalized linear models (i.e., "nls" and "glm" objects), on the other hand, rely on Taylor-series approximations for the standard errors used in forming the intervals. Approximate standard errors for the fitted values in linear mixed-effects models (i.e., "lme" objects) can also be computed; however, these rely on the approximate variance-covariance matrix of the fixed-effects estimates and often under estimate the true standard error. More accurate standard errors can be obtained using the parametric bootstrap; see bootMer for details.

For linear and nonlinear models, it is possible to request *adjusted* confidence or prediction intervals using the Bonferroni method (adjust = "Bonferroni") or Scheffe's method (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, calling plotFit on "lm" objects with interval = "confidence" and adjust = "Scheffe" will plot the Working-Hotelling band.

Value

If se.fit = FALSE, then predFit() returns a vector of predictions or a matrix of predictions and bounds with column names fit, lwr, and upr if interval is not "none". (This function is more so meant for internal use.)

If se.fit = TRUE, then a list with the following components is returned:

- fit a vector or matrix as described above;
- se.fit a vector containing the standard errors of the predicted means;
- residual.scale the residual standard deviations;
- df the residual degrees of freedom.

Examples

```
# A linear regression example (see ?datasets::cars)
cars.lm <- lm(dist ~ speed + I(speed^2), data = cars)
predFit(cars.lm, interval = "confidence")

# A nonlinear least squares example (see ?datasets::Puromycin)
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))
conc <- seq(from = 0.02, to = 1.10, length = 101)
pred <- predFit(Puro.nls, newdata = data.frame(conc), interval = "prediction")
plot(Puromycin2, ylim = c(min(pred[, "lwr"]), max(pred[, "upr"])))
lines(conc, pred[, "fit"], lwd = 2)
lines(conc, pred[, "lwr"], lty = 2)
lines(conc, pred[, "upr"], lty = 2)
```

---

whisky *Whisky data*

Description

The data give the proof (measured as twice the percentage of alcohol by volume, denoted 2ABV) of whiskey stored in a charred oak barrel against time in years.

Format

A data frame with 10 rows and 2 columns.

Details

- `age` The age of the whisky (years).
- `proof` The proof of the whisky (2ABV).

Source

R. Schoeneman, R. Dyer, and E. Earl. Analytical profile of straight bourbon whiskies. Journal of the Association of Official Analytical Chemists, 54:1247–1261, 1971.

Index