Package 'ciTools'

September 4, 2020

Type Package

Title Confidence or Prediction Intervals, Quantiles, and Probabilities for Statistical Models

Version 0.6.0

Maintainer John Haman <jhaman@ida.org>

Description Functions to append confidence intervals, prediction intervals, and other quantities of interest to data frames. All appended quantities are for the response variable, after conditioning on the model and covariates. This package has a data frame first syntax that allows for easy piping. Currently supported models include (log-) linear, (log-) linear mixed, generalized linear models, generalized linear mixed models, and accelerated failure time models.

Depends R (>= 3.4.0)

Imports arm, boot, dplyr, lme4, MASS, methods, stats, survival, utils

Suggests here, ggplot2, knitr, rmarkdown, SPREDA

VignetteBuilder rmarkdown, knitr

URL https://github.com/jthaman/ciTools

BugReports https://github.com/jthaman/ciTools/issues

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author John Haman [cre, aut],
Matthew Avery [aut],
Institute for Defense Analyses [cph]

Repository CRAN

Date/Publication 2020-09-04 21:22:16 UTC

R topics documented:

`add_ci` *Add Confidence Intervals for Fitted Values to Data Frames*

Description

This is a generic function to append confidence intervals for predictions of a model fit to a data frame. A confidence interval is generated for the fitted value of each observation in `df`. These confidence intervals are then appended to `df` and returned to the user as a data frame. The `fit` may be a linear, log-linear, linear mixed, generalized linear model, generalized linear mixed, or accelerated failure time model.

Usage

```
add_ci(df, fit, alpha = 0.05, names = NULL, yhatName = "pred", ...)
```

Arguments

df	A data frame of new data. df can be the original data or new data.
fit	An object of class lm, glm, lmerMod, glmerMod, or survreg. Predictions are made with this object.
alpha	A real number between 0 and 1. Controls the confidence level of the interval estimates.
names	NULL or character vector of length two. If NULL, confidence bounds automatically will be named by add_ci, otherwise, the lower confidence bound will be named names[1] and the upper confidence bound will be named names[2].
yhatName	A character vector of length one. Name of the vector of the predictions made for each observation in df
...	Additional arguments.

Details

For more specific information about the arguments that are applicable in each method, consult:

- add_ci.lm for linear model confidence intervals
- add_ci.glm for generalized linear model confidence intervals
- add_ci.lmerMod for linear mixed model confidence intervals
- add_ci.glmerMod for generalized linear mixed model confidence intervals
- add_ci.survreg for accelerated failure time confidence intervals

Note that add_ci calculates confidence intervals for *fitted values*, not model coefficients. For confidence intervals of model coefficients, see confint.

Value

A dataframe, df, with predicted values, upper and lower confidence bounds attached.

See Also

add_pi for prediction intervals, add_probs for response level probabilities, and add_quantile for quantiles of the conditional response distribution.

Examples

```
# Fit a linear model
fit <- lm(dist ~ speed, data = cars)
# Make a confidence interval for each observation in cars, and
# append to the data frame
add_ci(cars, fit)

# Make new data
new_data <- cars[sample(NROW(cars), 10), ]
add_ci(new_data, fit)
```

```
# Fit a Poisson model
fit2 <- glm(dist ~ speed, family = "poisson", data = cars)
# Append CIs
add_ci(cars, fit2)

# Fit a linear mixed model using lme4
fit3 <- lme4::lmer(Reaction ~ Days + (1|Subject), data = lme4::sleepstudy)
# Append CIs
# Generally, you should use more than 100 bootstrap replicates
add_ci(lme4::sleepstudy, fit3, nSims = 100)

# Fit a logistic model
fit4 <- glm(I(dist > 20) ~ speed, family = "binomial", data = cars)
# Append CIs
add_ci(cbind(cars, I(cars$dist > 20)), fit4)
```

`add_ci.glm` *Confidence Intervals for Generalized Linear Model Predictions*

Description

This function is one of the methods for `add_ci`, and is called automatically when `add_ci` is used on a `fit` of class `glm`. The default method calculates confidence intervals by making an interval on the scale of the linear predictor, then applying the inverse link function from the model fit to transform the linear level confidence intervals to the response level. Alternatively, confidence intervals may be calculated through a nonparametric bootstrap method.

Usage

```
## S3 method for class 'glm'
add_ci(
  df,
  fit,
  alpha = 0.05,
  names = NULL,
  yhatName = "pred",
  response = TRUE,
  type = "parametric",
  nSims = 2000,
  ...
)
```

Arguments

`df`	A data frame of new data.
`fit`	An object of class `glm`.

alpha	A real number between 0 and 1. Controls the confidence level of the interval estimates.
names	NULL or character vector of length two. If NULL, confidence bounds automatically will be named by add_ci, otherwise, the lower confidence bound will be named names[1] and the upper confidence bound will be named names[2].
yhatName	A character vector of length one. Name of the vector of predictions made for each observation in df
response	A logical. The default is TRUE. If TRUE, the confidence intervals will be determined for the expected response; if FALSE, confidence intervals will be made on the scale of the linear predictor.
type	A character vector of length one. Must be type = "parametric" or type = "boot". type determines the method used to compute the confidence intervals.
nSims	An integer. Number of simulations to perform if the bootstrap method is used.
...	Additional arguments.

Value

A dataframe, df, with predicted values, upper and lower confidence bounds attached.

See Also

add_pi.glm for prediction intervals for glm objects, add_probs.glm for conditional probabilities of glm objects, and add_quantile.glm for response quantiles of glm objects.

Examples

```
# Poisson regression
fit <- glm(dist ~ speed, data = cars, family = "poisson")
add_ci(cars, fit)
# Try a different confidence level
add_ci(cars, fit, alpha = 0.5)
# Add custom names to the confidence bounds (may be useful for plotting)
add_ci(cars, fit, alpha = 0.5, names = c("lwr", "upr"))

# Logistic regression
fit2 <- glm(I(dist > 30) ~ speed, data = cars, family = "binomial")
dat <- cbind(cars, I(cars$dist > 30))
# Form 95% confidence intervals for the fit:
add_ci(dat, fit2)
# Form 50% confidence intervals for the fit:
add_ci(dat, fit2, alpha = 0.5)
# Make confidence intervals on the scale of the linear predictor
add_ci(dat, fit2, alpha = 0.5, response = FALSE)
# Add custom names to the confidence bounds
add_ci(dat, fit2, alpha = 0.5, names = c("lwr", "upr"))
```

`add_ci.glmerMod` *Confidence Intervals for Generalized Linear Mixed Model Predictions*

Description

This function is one of the methods for `add_ci`, and is called automatically when `add_ci` is used on a `fit` of class `glmerMod`.

Usage

```
## S3 method for class 'glmerMod'
add_ci(
  df,
  fit,
  alpha = 0.05,
  names = NULL,
  yhatName = "pred",
  response = TRUE,
  type = "boot",
  includeRanef = TRUE,
  nSims = 500,
  ...
)
```

Arguments

`df`	A data frame of new data.
`fit`	An object of class `glmerMod`.
`alpha`	A real number between 0 and 1. Controls the confidence level of the interval estimates.
`names`	`NULL` or character vector of length two. If `NULL`, confidence bounds automatically will be named by `add_ci`, otherwise, the lower confidence bound will be named `names[1]` and the upper confidence bound will be named `names[2]`.
`yhatName`	`NULL` or a string. Name of the predictions vector. If `NULL`, the predictions will be named `pred`.
`response`	A logical. The default is `TRUE`. If `TRUE`, the confidence intervals will be determined for the expected response; if `FALSE`, confidence intervals will be made on the scale of the linear predictor.
`type`	A string. If `type == "boot"` then bootstrap intervals are formed. If `type == "parametric"` then parametric intervals are formed. Currently only bootstrap intervals are supported.
`includeRanef`	A logical. Default is `TRUE`. Set whether the predictions and intervals should be made conditional on the random effects. If `FALSE`, random effects will not be included.
`nSims`	A positive integer. Controls the number of bootstrap replicates if `type = "boot"`.
`...`	Additional arguments.

Details

The default and recommended method is bootstrap. The bootstrap method can handle many types of models and we find it to be generally reliable and robust as it is built on the `bootMer` function from `lme4`. This function is experimental.

If `IncludeRanef` is False, random slopes and intercepts are set to 0. Unlike in 'lmer' fits, settings random effects to 0 does not mean they are marginalized out. Consider generalized estimating equations if this is desired.

Value

A dataframe, `df`, with predicted values, upper and lower confidence bounds attached.

References

For general information about GLMMs http://bbolker.github.io/mixedmodels-misc/glmmFAQ.html

See Also

`add_pi.glmerMod` for prediction intervals of `glmerMod` objects, `add_probs.glmerMod` for conditional probabilities of `glmerMod` objects, and `add_quantile.glmerMod` for response quantiles of `glmerMod` objects.

Examples

```
n <- 300
x <- runif(n)
f <- factor(sample(1:5, size = n, replace = TRUE))
y <- rpois(n, lambda = exp(1 - 0.05 * x * as.numeric(f) + 2 * as.numeric(f)))
df <- data.frame(x = x, f = f, y = y)
fit <- lme4::glmer(y ~ (1+x|f), data=df, family = "poisson")

## Not run: add_ci(df, fit, names = c("lcb", "ucb"), nSims = 300)
```

`add_ci.lm` *Confidence Intervals for Linear Model Predictions*

Description

This function is one of the methods in `add_ci` and automatically is called when an object of class `lm` is passed to `add_ci`.

Usage

```
## S3 method for class 'lm'
add_ci(
  df,
  fit,
  alpha = 0.05,
  names = NULL,
  yhatName = "pred",
  log_response = FALSE,
  ...
)
```

Arguments

df	A data frame.
fit	An object of class lm. Predictions are made with this object.
alpha	A real number between 0 and 1. Controls the confidence level of the interval estimates.
names	NULL or character vector of length two. If NULL, confidence bounds automatically will be named by add_ci, otherwise, the lower confidence bound will be named names[1] and the upper confidence bound will be named names[2].
yhatName	A string. Name of the vector of the predictions made for each observation in df
log_response	Logical. Default is FALSE. If TRUE, confidence intervals will be generated for the *response level* of a log-linear model: $log(Y) = X\beta + \epsilon$.
...	Additional arguments.

Details

Confidence intervals for lm objects are calculated parametrically. This function is essentially a wrapper for predict(fit,df,interval = "confidence") if fit is a linear model. If log_response = TRUE, confidence intervals for the response are calculated using Wald's Method. See Meeker and Escobar (1998) for details.

Value

A dataframe, df, with predicted values, upper and lower confidence bounds attached.

See Also

add_pi.lm for prediction intervals for lm objects, add_probs.lm for conditional probabilities of lm objects, and add_quantile.lm for response quantiles of lm objects.

Examples

```
# Fit a linear model
fit <- lm(dist ~ speed, data = cars)
# Get fitted values for each observation in cars, and append
```

```
# confidence intervals
add_ci(cars, fit)
# Try a different confidence level
add_ci(cars, fit, alpha = 0.5)
# Try custom names for the confidence bounds
add_ci(cars, fit, alpha = 0.5, names = c("lwr", "upr"))
```

`add_ci.lmerMod` *Confidence Intervals for Linear Mixed Model Predictions*

Description

This function is one of the methods for `add_ci`, and is called automatically when `add_ci` is used on a `fit` of class `lmerMod`. It is recommended that one use parametric confidence intervals when modeling with a random intercept linear mixed model (i.e. a fit with a formula such as `lmer(y ~ x + (1|group))`). Otherwise, confidence intervals may be bootstrapped via `lme4::bootMer`.

Usage

```
## S3 method for class 'lmerMod'
add_ci(
  df,
  fit,
  alpha = 0.05,
  names = NULL,
  yhatName = "pred",
  type = "boot",
  includeRanef = TRUE,
  nSims = 500,
  ...
)
```

Arguments

`df`	A data frame of new data.
`fit`	An object of class `lmerMod`.
`alpha`	A real number between 0 and 1. Controls the confidence level of the interval estimates.
`names`	`NULL` or character vector of length two. If `NULL`, confidence bounds automatically will be named by `add_ci`, otherwise, the lower confidence bound will be named `names[1]` and the upper confidence bound will be named `names[2]`.
`yhatName`	A string. Name of the predictions vector.
`type`	A string. Must be `"parametric"` or `"boot"`, If `type = "boot"`, then `add_ci` calls `lme4::bootMer` to calculate the confidence intervals. This method may be time consuming, but is applicable with random slope and random intercept models. The parametric method is fast, but currently only works well for random intercept models.

includeRanef	A logical. Default is TRUE. Set whether the predictions and intervals should be made conditional on the random effects. If FALSE, random effects will not be included.
nSims	A positive integer. Controls the number of bootstrap replicates if type = "boot".
...	Additional arguments.

Details

Bootstrapped intervals are slower to compute, but they are the recommended method when working with any linear mixed models more complicated than the random intercept model.

Value

A dataframe, df, with predicted values, upper and lower confidence bounds attached.

See Also

add_pi.lmerMod for prediction intervals of lmerMod objects, add_probs.lmerMod for conditional probabilities of lmerMod objects, and add_quantile.lmerMod for response quantiles of lmerMod objects.

Examples

```
## Not run:
dat <- lme4::sleepstudy
# Fit a linear mixed model (random intercept model)
fit <- lme4::lmer(Reaction ~ Days + (1|Subject), data = lme4::sleepstudy)
# Get the fitted values for each observation in dat, and
# append CIs for those fitted values to dat
add_ci(dat, fit, alpha = 0.5)
# Try the parametric bootstrap method, and make prediction at the population level
add_ci(dat, fit, alpha = 0.5, type = "boot", includeRanef = FALSE, nSims = 100)

## End(Not run)
```

---

add_ci.negbin *Confidence Intervals for Negative Binomial Linear Model Predictions*

Description

This function is one of the methods for add_ci, and is called automatically when add_ci is used on a fit of class negbin.

Usage

```
## S3 method for class 'negbin'
add_ci(
  df,
  fit,
  alpha = 0.05,
  names = NULL,
  yhatName = "pred",
  response = TRUE,
  type = "parametric",
  nSims = 2000,
  ...
)
```

Arguments

df	A data frame of new data.
fit	An object of class negbin.
alpha	A real number between 0 and 1. Controls the confidence level of the interval estimates.
names	NULL or character vector of length two. If NULL, confidence bounds automatically will be named by add_ci, otherwise, the lower confidence bound will be named names[1] and the upper confidence bound will be named names[2].
yhatName	A string. Name of the vector of predictions made for each observation in df
response	A logical. The default is TRUE. If TRUE, the confidence intervals will be determined for the expected response; if FALSE, confidence intervals will be made on the scale of the linear predictor.
type	A string. Must be type = "parametric" or type = "boot". type determines the method used to compute the confidence intervals.
nSims	An integer. Number of simulations to perform if the bootstrap method is used.
...	Additional arguments.

Details

The default link function is log-link. Confidence Intervals are determined by making an interval on the scale of the linear predictor, then applying the inverse link function from the model fit to transform the linear level confidence intervals to the response level. Alternatively, bootstrap confidence intervals may be formed. The bootstrap intervals are formed by first resampling cases from the data frame used to calculate fit, then bias corrected and accelerated intervals are calculated. See boot::boot.ci for more details.

Value

A dataframe, df, with predicted values, upper and lower confidence bounds attached.

See Also

add_pi.negbin for prediction intervals for negbin objects, add_probs.negbin for conditional probabilities of negbin objects, and add_quantile.negbin for response quantiles of negbin objects.

Examples

```
x1 <- rnorm(300, mean = 1)
y <- MASS::rnegbin(n = 300, mu = exp(5 + 0.5 * x1), theta = 2)
df <- data.frame(x1 = x1, y = y)
fit <- MASS::glm.nb(y ~ x1, data = df)
df <- df[sample(100),]
add_ci(df, fit, names = c("lcb", "ucb"))
```

---

add_ci.survreg *Confidence Intervals for the Mean Survival Time of Accelerated Failure Time Models.*

Description

This function is one of the methods for add_ci, and is called automatically when add_ci is used on a fit of class survreg.

Usage

```
## S3 method for class 'survreg'
add_ci(df, fit, alpha = 0.1, names = NULL, yhatName = "mean_pred", ...)
```

Arguments

df	A data frame of new data on which to form predictions and confidence intervals.
fit	An object of class survreg. Predictions are made with this object.
alpha	A number between 0 and 1. 1 - alpha is the confidence level of the intervals.
names	NULL or a string of length 2. If NULL, quantiles automatically will be named by add_quantile, otherwise, they will be named names.
yhatName	A string. Name of the vector of predictions. The default name is mean_pred.
...	Additional arguments.

Details

add_ci.survreg calculates confidence intervals for the mean survival time of several accelerated failure time (AFT) models including exponential, lognormal, weibull, and loglogistic models. AFT models must be fit with the survreg function in the survival package. Confidence intervals are formed parametrically via the Delta method.

add_ci.survreg will compute confidence intervals for the following mean survival time point estimates:

Exponential: $E[Y|X] = \exp X\beta$

Weibull: $E[Y|X] = \exp X\beta\Gamma(1+\sigma)$

Lognormal: $E[Y|X] = \exp X\beta + \frac{\sigma^2}{2}$

Loglogistic: $E[Y|X] = \exp X\beta\Gamma(1+\sigma)(1-\sigma)$

Traditionally, survival time predictions are made with the median survival time. For forming confidence intervals for the median survival time (or any quantile of the survival time distribution), see add_quantile.survreg.

Note: The expected survival time of a loglogistic model with scale >= 1 does not exist. Otherwise, expected survival times exist for each of the four AFT models considered in add.ci_survreg.

Note: Due to a limitation, the Surv object must be specified in survreg function call. See the examples section for one way to do this.

Note: add_ci.survreg cannot inspect the convergence of fit. Poor maximum likelihood estimates will result in poor confidence intervals. Inspect any warning messages given from survreg.

Value

A dataframe, df, with predicted expected values and level *1 - alpha* level confidence levels attached.

References

For descriptions of the log-location scale models supported: Meeker, William Q., and Luis A. Escobar. Statistical methods for reliability data. John Wiley & Sons, 2014. (Chapter 4)

For a description of the multivariate Delta method: Meeker, William Q., and Luis A. Escobar. Statistical methods for reliability data. John Wiley & Sons, 2014. (Appendix B.2)

See Also

add_quantile.survreg for quantiles of the survival time distribution of survreg objects, add_pi.survreg for prediction intervals of survreg objects, and add_probs.survreg for survival probabilities of survreg objects.

Examples

```
## Define a data set.
df <- survival::stanford2
## remove a covariate with missing values.
df <- df[, 1:4]
## next, create the Surv object inside the survreg call:
fit <- survival::survreg(survival::Surv(time, status) ~ age + I(age^2),
                         data = df, dist = "lognormal")
add_ci(df, fit, alpha = 0.1, names = c("lwr", "upr"))

## Try a different model:
fit2 <- survival::survreg(survival::Surv(time, status) ~ age + I(age^2),
                          data = df, dist = "weibull")
```

```
add_ci(df, fit2, alpha = 0.1, names = c("lwr", "upr"))
```

---

add_pi *Add Prediction Intervals to Data Frames*

Description

This is a generic function to append prediction intervals to a data frame. A prediction interval is made for each observation in `df` with respect to the model `fit`. These intervals are then appended to `df` and returned to the user as a data frame. `fit` can be a linear, log-linear, linear mixed, generalized linear, generalized linear mixed, or accelerated failure time model.

Usage

```
add_pi(df, fit, alpha = 0.05, names = NULL, yhatName = "pred", ...)
```

Arguments

`df`	A data frame of new data.
`fit`	An object of class `lm`, `glm`, or `lmerMod`. Predictions are made with this object.
`alpha`	A real number between 0 and 1. Controls the confidence level of the interval estimates.
`names`	`NULL` or character vector of length two. If `NULL`, prediction bounds automatically will be named by `add_pi`, otherwise, the lower prediction bound will be named `piNames[1]` and the upper prediction bound will be named `piNames[2]`.
`yhatName`	A string. Name of the predictions vector.
`...`	Additional arguments

Details

For more specific information about the arguments that are applicable in each method, consult:

- `add_pi.lm` for linear regression prediction intervals
- `add_pi.glm` for generalized linear regression prediction intervals
- `add_pi.lmerMod` for linear mixed models prediction intervals
- `add_pi.glmerMod` for generalized linear mixed model prediction intervals
- `add_pi.survreg` for accelerated failure time model prediction intervals

Value

A dataframe, `df`, with predicted values, upper and lower prediction bounds attached.

See Also

add_ci for confidence intervals, add_probs for response level probabilities, and add_quantile for quantiles of the conditional response distribution.

Examples

```
# Fit a linear model
fit <- lm(dist ~ speed, data = cars)
# Define some new data
new_data <- cars[sample(NROW(cars), 10), ]
# Add fitted values and prediction intervals to new_data
add_pi(new_data, fit)

# Fit a Poisson model
fit2 <- glm(dist ~ speed, family = "poisson", data = cars)
# Add approximate prediction intervals to the fitted values of
# new_data
add_pi(new_data, fit2)

# Fit a linear mixed model
fit3 <- lme4::lmer(Reaction ~ Days + (1|Subject), data = lme4::sleepstudy)
# Add parametric prediction intervals for the fitted values to the
# original data
add_pi(lme4::sleepstudy, fit3, type = "parametric")
```

---

add_pi.glm *Prediction Intervals for Generalized Linear Models*

Description

This function is one of the methods for add_pi, and is called automatically when add_pi is used on a fit of class glm.

Usage

```
## S3 method for class 'glm'
add_pi(
  df,
  fit,
  alpha = 0.05,
  names = NULL,
  yhatName = "pred",
  nSims = 2000,
  ...
)
```

Arguments

df	A data frame of new data.
fit	An object of class glm.
alpha	A real number between 0 and 1. Controls the confidence level of the interval estimates.
names	NULL or character vector of length two. If NULL, prediction bounds automatically will be named by add_pi, otherwise, the lower prediction bound will be named names[1] and the upper prediction bound will be named names[2].
yhatName	A string. Name of the predictions vector.
nSims	A positive integer. Determines the number of simulations to run.
...	Additional arguments.

Details

Prediction intervals are generated through simulation with the aid arm::sim, which simulates the uncertainty in the regression coefficients. At the moment, only prediction intervals for Poisson, Quasipoisson, Gaussian, and Gamma GLMs are supported. Note that if the response is count data, prediction intervals are only approximate. Simulation from the QuasiPoisson model is performed with the negative binomial distribution, see Gelman and Hill (2007).

Value

A dataframe, df, with predicted values, upper and lower prediction bounds attached.

See Also

add_ci.glm for confidence intervals for glm objects, add_probs.glm for conditional probabilities of glm objects, and add_quantile.glm for response quantiles of glm objects.

Examples

```
# Fit a Poisson model
fit <- glm(dist ~ speed, data = cars, family = "poisson")
# Add prediction intervals and fitted values to the original data frame
add_pi(cars, fit)
# Try a different confidence level
add_pi(cars, fit, alpha = 0.5)
# Try custom names for the prediction bounds (may be useful for plotting)
add_pi(cars, fit, alpha = 0.5, names = c("lwr", "upr"))
```

add_pi.glmerMod *Prediction Intervals for Generalized Linear Mixed Model Predictions*

Description

This function is one of the methods for add_pi, and is called automatically when add_pi is used on a fit of class glmerMod. This function is experimental.

Usage

```
## S3 method for class 'glmerMod'
add_pi(
  df,
  fit,
  alpha = 0.05,
  names = NULL,
  yhatName = "pred",
  type = "boot",
  includeRanef = TRUE,
  nSims = 10000,
  ...
)
```

Arguments

df	A data frame of new data.
fit	An object of class glmerMod.
alpha	A real number between 0 and 1. Controls the confidence level of the interval estimates.
names	NULL or character vector of length two. If NULL, prediction bounds automatically will be named by add_pi, otherwise, the lower prediction bound will be named names[1] and the upper prediction bound will be named names[2].
yhatName	NULL or a string. The name of the predictions vector.
type	A string. Must be "boot", If type = "boot", then add_ci calls lme4::bootMer to calculate the confidence intervals.
includeRanef	A logical. Default is TRUE. Set whether the predictions and intervals should be made conditional on the random effects. If FALSE, random effects will not be included.
nSims	A positive integer. Controls the number of bootstrap replicates.
...	Additional arguments.

Details

Prediction intervals are approximate and determined by simulation through the `simulate` function distributed with `lme4`.

If `IncludeRanef` is False, random slopes and intercepts are set to 0. Unlike in 'lmer' fits, settings random effects to 0 does not mean they are marginalized out. Consider generalized estimating equations if this is desired.

Value

A dataframe, `df`, with predicted values, upper and lower prediction bounds attached.

See Also

`add_ci.glmerMod` for confidence intervals of `glmerMod` objects, `add_probs.glmerMod` for conditional probabilities of `glmerMod` objects, and `add_quantile.glmerMod` for response quantiles of `glmerMod` objects.

Examples

```
n <- 300
x <- runif(n)
f <- factor(sample(1:5, size = n, replace = TRUE))
y <- rpois(n, lambda = exp(1 - 0.05 * x * as.numeric(f) + 2 * as.numeric(f)))
df <- data.frame(x = x, f = f, y = y)
fit <- lme4::glmer(y ~ (1+x|f), data=df, family = "poisson")

add_pi(df, fit, names = c("LPB", "UPB"), nSims = 500)
```

`add_pi.lm` *Prediction Intervals for Linear Model Predictions*

Description

This function is one of the methods for `add_pi` and is automatically called when an object of class `lm` is passed to to `add_pi`.

Usage

```
## S3 method for class 'lm'
add_pi(
  df,
  fit,
  alpha = 0.05,
  names = NULL,
  yhatName = "pred",
  log_response = FALSE,
  ...
)
```

Arguments

df	A data frame of new data.
fit	An object of class lm. Predictions are made with this object.
alpha	A real number between 0 and 1. Controls the confidence level of the interval estimates.
names	NULL or character vector of length two. If NULL, prediction bounds automatically will be named by add_pi, otherwise, the lower prediction bound will be named names[1] and the upper prediction bound will be named names[2].
yhatName	A string. Name of the predictions vector.
log_response	A logical. If TRUE, prediction intervals will be generated at the *response level* of a log-linear model: $\log(Y) = X\beta + \epsilon$. Again, these intervals will be on the scale of the original response, Y.
...	Additional arguments.

Details

Prediction intervals for lm objects are calculated parametrically. This function is essentially just a wrapper for predict(fit,df,interval = "prediction") if fit is a linear model. If log_response = TRUE, prediction intervals for the response are calculated parametrically, then the exponential function is applied to transform them to the original scale.

Value

A dataframe, df, with predicted values, upper and lower prediction bounds attached.

See Also

add_ci.lm for confidence intervals for lm objects. add_probs.lm for conditional probabilities of lm objects, and add_quantile.lm for response quantiles of lm objects.

Examples

```
# Fit a linear model
fit <- lm(dist ~ speed, data = cars)
# Add prediction intervals and fitted values to the original data
add_pi(cars, fit)

# Try to add predictions to a data frame of new data
new_data <- cars[sample(NROW(cars), 10), ]
add_pi(new_data, fit)

# Try a different confidence level
add_pi(cars, fit, alpha = 0.5)

# Add custom names to the prediction bounds.
add_pi(cars, fit, alpha = 0.5, names = c("lwr", "upr"))
```

add_pi.lmerMod *Prediction Intervals for Linear Mixed Model Fitted Values*

Description

This function is one of the methods in add_pi, and is called automatically when add_pi is used on a fit of class lmerMod.

Usage

```
## S3 method for class 'lmerMod'
add_pi(
  df,
  fit,
  alpha = 0.05,
  names = NULL,
  yhatName = "pred",
  type = "parametric",
  includeRanef = TRUE,
  log_response = FALSE,
  nSims = 10000,
  ...
)
```

Arguments

df	A data frame of new data
fit	An object of class lmerMod.
alpha	A real number between 0 and 1. Controls the confidence level of the interval estimates.
names	NULL or character vector of length two. If NULL, prediction bounds automatically will be named by add_pi, otherwise, the lower prediction bound will be named names[1] and the upper prediction bound will be named names[2].
yhatName	A string. Name of the predictions vector.
type	A string, either "parametric" or "boot". Determines the method used to calculate the prediction intervals.
includeRanef	A logical. Set whether the predictions and intervals should be conditioned on the random effects. If FALSE, random effects will not be included.
log_response	A logical, indicating if the response is on log scale in the model fit. If TRUE, prediction intervals will be returned on the response scale.
nSims	A positive integer. If type = "boot", nSims will determine the number of bootstrap simulations to perform.
...	Additional arguments.

Details

It is recommended that one use parametric prediction intervals when modeling with a random intercept linear mixed model. Otherwise, prediction intervals may be simulated via a parametric bootstrap using the function `lme4.simulate()`.

Value

A dataframe, `df`, with predicted values, upper and lower prediction bounds attached.

See Also

`add_ci.lmerMod` for confidence intervals for `lmerMod` objects, `add_probs.lmerMod` for conditional probabilities of `lmerMod` objects, and `add_quantile.lmerMod` for response quantiles of `lmerMod` objects.

Examples

```
dat <- lme4::sleepstudy
# Fit a (random intercept) linear mixed model
fit <- lme4::lmer(Reaction ~ Days + (1|Subject), data = lme4::sleepstudy)
# Add 50% prediction intervals to the original data using the default
# method.
add_pi(dat, fit, alpha = 0.5)

# Add 50% prediction intervals to the original data using the
# parametric bootstrap method. Form prediction intervals at the population
# level (unconditional on the random effects).
add_pi(dat, fit, alpha = 0.5, type = "boot", includeRanef = FALSE)
```

---

add_pi.negbin *Prediction Intervals for Negative Binomial Linear Models*

Description

This function is one of the methods for `add_pi`, and is called automatically when `add_pi` is used on a `fit` of class `negbin`.

Usage

```
## S3 method for class 'negbin'
add_pi(
  df,
  fit,
  alpha = 0.05,
  names = NULL,
  yhatName = "pred",
  nSims = 2000,
  ...
)
```

Arguments

df	A data frame of new data.
fit	An object of class negbin.
alpha	A real number between 0 and 1. Controls the confidence level of the interval estimates.
names	NULL or character vector of length two. If NULL, prediction bounds automatically will be named by add_pi, otherwise, the lower prediction bound will be named names[1] and the upper prediction bound will be named names[2].
yhatName	A string. Name of the predictions vector.
nSims	A positive integer. Determines the number of simulations to run.
...	Additional arguments.

Details

Prediction intervals for negative binomial fits are formed through a two part simulation scheme:

1. Model coefficients are generated through a parametric bootstrap procedure that simulates the uncertainty in the regression coefficients.

2. Random draws from the negative binomial distribution are taken with a mean that varies based on the model coefficients determined in step (1) and over-dispersion parameter that is taken from the original fitted model.

Quantiles of the simulated responses are taken at the end to produce intervals of the desired level.

Value

A dataframe, df, with predicted values, upper and lower prediction bounds attached.

See Also

add_ci.negbin for confidence intervals for negbin objects, add_probs.negbin for conditional probabilities of negbin objects, and add_quantile.negbin for response quantiles of negbin objects.

Examples

```
x1 <- rnorm(100, mean = 1)
y <- MASS::rnegbin(n = 100, mu = exp(1 + x1), theta = 5)
df <- data.frame(x1 = x1, y = y)
fit <- MASS::glm.nb(y ~ x1, data = df)
add_pi(df, fit, names = c("lpb", "upb"))
```

add_pi.survreg *Prediction Intervals for Accelerated Failure Time Models*

Description

This function is one of the methods for add_pi, and is called automatically when add_pi is used on a fit of class survreg.

Usage

```
## S3 method for class 'survreg'
add_pi(
  df,
  fit,
  alpha = 0.05,
  names = NULL,
  yhatName = "median_pred",
  nSims = 10000,
  method = "naive",
  ...
)
```

Arguments

df	A data frame of new data.
fit	An object of class survreg.
alpha	A real number between 0 and 1. Controls the confidence level of the interval estimates.
names	NULL or character vector of length two. If NULL, prediction bounds automatically will be named by add_pi, otherwise, the lower prediction bound will be named names[1] and the upper prediction bound will be named names[2].
yhatName	A string. Name of the predictions vector.
nSims	A positive integer. Determines the number of bootstrap replicates if method = "boot".
method	A string. Determines the method used to calculate prediction intervals. Must be one of either "naive" or "boot".
...	Additional arguments.

Details

add_pi.survreg creates prediction intervals for the survival time T conditioned on the covariates of the survreg model. In simple terms, this function calculates error bounds within which one can expect to observe a new survival time. Like other parametric survival methods in ciTools, prediction intervals are limited to unweighted lognormal, exponential, weibull, and loglogistic AFT models.

Two methods are available for creating prediction intervals, the "naive" method (Meeker and Escobar, chapter 8) and a simulation method that implements a parametric bootstrap routine. The "naive" method calculates quantiles of the fitted survival time distribution to determine prediction intervals. The parametric bootstrap method simulates new survival times from the conditional survival time distribution, taking into account the uncertainty in the regression coefficients. The bootstrap method is similar to the one implemented in `add_pi.glm`.

Note: Due to a limitation, the `Surv` object must be specified in `survreg` function call. See the examples section for one way to do this.

Note: `add_pi.survreg` cannot inspect the convergence of `fit`. Poor maximum likelihood estimates will result in poor prediction intervals. Inspect any warning messages given from `survreg`.

Value

A dataframe, `df`, with predicted medians, upper and lower prediction bounds attached.

References

For a discussion prediction intervals of accelerated failure time models: Meeker, William Q., and Luis A. Escobar. Statistical methods for reliability data. John Wiley & Sons, 2014. (Chapter 8)

See Also

`add_ci.survreg` for confidence intervals for `survreg` objects, `add_probs.survreg` for conditional survival probabilities of `survreg` objects, and `add_quantile.survreg` for survival time quantiles of `survreg` objects.

Examples

```
## Define a data set.
df <- survival::stanford2
## remove a covariate with missing values.
df <- df[, 1:4]
## next, create the Surv object inside the survreg call:
fit <- survival::survreg(survival::Surv(time, status) ~ age + I(age^2),
                         data = df, dist = "lognormal")
add_pi(df, fit, alpha = 0.1, names = c("lwr", "upr"))

## Try a different model:
fit2 <- survival::survreg(survival::Surv(time, status) ~ age + I(age^2),
                          data = df, dist = "weibull")
add_pi(df, fit2, alpha = 0.1, names = c("lwr", "upr"))
```

add_probs *Add Regression Probabilities to Data Frames*

Description

This is a generic function to append response level probabilities to a data frame. A response level probability (conditioned on the model and covariates), such as $Pr(Response|Covariates < 10)$, is generated for the fitted value of each observation in `df`. These probabilities are then appended to `df` and returned to the user as a data frame.

Usage

```
add_probs(df, fit, q, name = NULL, yhatName = "pred", comparison, ...)
```

Arguments

`df`	A data frame of new data.
`fit`	An object of class `lm`, `glm`, or `lmerMod`. Predictions are made with this object.
`q`	A real number. A quantile of the conditional response distribution.
`name`	`NULL` or character vector of length one. If `NULL`, probabilities automatically will be named by `add_probs`, otherwise, the probabilities will be named `name` in the returned data frame.
`yhatName`	A character vector of length one. Names of the
`comparison`	A string. If `comparison = "<"`, then $Pr(Y\|x < q)$ is calculated for each observation in `df`. Default is `"<"`. Must be `"<"` or `">"` for objects of class `lm` or `lmerMod`. If `fit` is a `glm`, then `comparison` also may be `"<="` , `">="` , or `"="`.
`...`	Additional arguments

Details

For more specific information about the arguments that are useful in each method, consult:

- `add_probs.lm` for linear regression response probabilities
- `add_probs.glm` for generalized linear regression response probabilities
- `add_probs.lmerMod` for linear mixed models response probabilities
- `add_probs.glmerMod` for generalized linear mixed model response probabilities
- `add_probs.survreg` for accelerated failure time model response probabilities

Note: Except in `add_probs.survreg`, the probabilities calculated by `add_probs` are on the distribution of $Y|x$, not $E[Y|x]$. That is, they use the same distribution from which a prediction interval is determined, not the distribution that determines a confidence interval. `add_probs.survreg` is an exception to this pattern so that users of accelerated failure time models can obtain estimates of the survivor function.

Value

A dataframe, df, with predicted values and probabilities attached.

See Also

add_ci for confidence intervals, add_quantile for response level quantiles, and add_pi for prediction intervals.

Examples

```
# Define a model
fit <- lm(dist ~ speed, data = cars)

# Calculate the probability that the probability that a new
# dist is less than 20 (Given the model).
add_probs(cars, fit, q = 20)

# Calculate the probability that a new
# dist is greater than 20 (Given the model).
add_probs(cars, fit, q = 20, comparison = ">")

# Try a different model fit.
fit2 <- glm(dist ~ speed, family = "poisson", data = cars)
add_probs(cars, fit2, q = 20)

# Try a different model fit.
fit3 <- lme4::lmer(Reaction ~ Days + (1|Subject), data = lme4::sleepstudy)
add_probs(lme4::sleepstudy, fit3, q = 300, type = "parametric")

# As above, but do not condition on the random effects.
add_probs(lme4::sleepstudy, fit3, q = 300, type = "parametric", includeRanef = FALSE)
```

---

add_probs.glm *Response Probabilities for Generalized Linear Models*

Description

This is the method add_probs uses if the model fit is an object of class glm. Probabilities are determined through simulation, using the same method as add_pi.glm. Currently, only logistic, Poisson, Quasipoisson, and Gamma models are supported.

Usage

```
## S3 method for class 'glm'
add_probs(
  df,
  fit,
  q,
```

```
  name = NULL,
  yhatName = "pred",
  comparison = "<",
  nSims = 2000,
  ...
)
```

Arguments

df	A data frame of new data.
fit	An object of class glm. Predictions are made with this object.
q	A double. A quantile of the response distribution.
name	NULL or a string. If NULL, probabilities automatically will be named by add_probs(), otherwise, the probabilities will be named name in the returned dataframe.
yhatName	A string. Name of the vector of predictions.
comparison	A character vector of length one. If comparison = "<", then $Pr(Y\|X < q)$ is calculated. Any comparison is allowed in Poisson regression, but only certain comparisons may be made in Logistic regression. See the Details section.
nSims	A positive integer. Controls the number of simulated draws to make if the model is Poisson.
...	Additional arguments.

Details

Any of the five comparisons may be made for a Poisson, quasipoisson, or Gamma model: comparison = "<", ">", "=", "<=", or ">=". For logistic regression, the comparison statement must be equivalent to $Pr(Y|x = 0)$ or $Pr(Y|x = 1)$.

If add_probs is called on a Poisson, quasiPoisson or Gamma model, a simulation is performed using arm::sim.

If add_probs is called on a logistic model, the fitted probabilities are used directly (no simulation is required).

If add_probs is called on a Gaussian GLM, the returned probabilities are identical to those given by add_probs.lm. In this case, the comparisons < and <= are identical (likewise for > and >=). If the comparison = is used in the Gaussian GLM, an informative error is returned.

Value

A dataframe, df, with predicted values and probabilities attached.

See Also

add_ci.glm for confidence intervals for glm objects, add_pi.glm for prediction intervals of glm objects, and add_quantile.glm for response quantiles of glm objects.

Examples

```
# Fit a Poisson model
fit <- glm(dist ~ speed, data = cars, family = "poisson")

# Determine the probability that a new dist is less than 20, given
# the Poisson model.
add_probs(cars, fit, q = 20)

# Determine the probability that a new dist is greater than 20,
# given the Poisson model.
add_probs(cars, fit, q = 30, comparison = ">")

# Determine the probability that a new dist is greater than or
# equal to 20, given the Poisson model.
add_probs(cars, fit, q = 30, comparison = ">=")

# Fit a logistic model
fit2 <- glm(I(dist > 30) ~ speed, data = cars, family = "binomial")
add_probs(cars, fit2, q = 0, comparison = "=")
add_probs(cars, fit2, q = 1, comparison = "=")
```

---

add_probs.glmerMod *Response Probabilities for Generalized Linear Mixed Model Predictions*

Description

This function is one of the methods for add_probs, and is called automatically when add_probs is used on a fit of class glmerMod. Probabilities are approximate and determined via a simulation. This function is experimental.

Usage

```
## S3 method for class 'glmerMod'
add_probs(
  df,
  fit,
  q,
  name = NULL,
  yhatName = "pred",
  comparison = "<",
  type = "boot",
  includeRanef = TRUE,
  nSims = 10000,
  ...
)
```

Arguments

df	A data frame of new data.
fit	An object of class glmerMod.
q	A double. A quantile of the response distribution.
name	NULL or character vector of length one. If NULL, response probabilities automatically will be named by add_probs,
yhatName	NULL or a string. Name of the predictions vector.
comparison	A string. If comparison = "<", then $Pr(Y\|x < q)$ is calculated for each observation in df. Default is "<". Must be "<" or ">" for objects of class lm or lmerMod. If fit is a glm or glmerMod, then comparison also may be "<=" , ">=" , or "=".
type	A string. Must be "boot", If type = "boot", then add_ci calls lme4::simulate to calculate the probabilities.
includeRanef	A logical. Default is TRUE. Set whether the predictions and intervals should be made conditional on the random effects. If FALSE, random effects will not be included.
nSims	A positive integer. Controls the number of bootstrap replicates if type = "boot".
...	Additional arguments.

Details

If IncludeRanef is False, random slopes and intercepts are set to 0. Unlike in 'lmer' fits, settings random effects to 0 does not mean they are marginalized out. Consider generalized estimating equations if this is desired.

Value

A dataframe, df, with predicted values and estimated probabilities attached.

See Also

add_pi.glmerMod for prediction intervals of glmerMod objects, add_ci.glmerMod for confidence intervals of glmerMod objects, and add_quantile.glmerMod for response quantiles of glmerMod objects.

Examples

```
n <- 300
x <- runif(n)
f <- factor(sample(1:5, size = n, replace = TRUE))
y <- rpois(n, lambda = exp(1 - 0.05 * x * as.numeric(f) + 2 * as.numeric(f)))
df <- data.frame(x = x, f = f, y = y)
fit <- lme4::glmer(y ~ (1+x|f), data=df, family = "poisson")

add_probs(df, fit, name = "p0.6", q = 0.6, nSims = 500)
```

add_probs.lm *Response Level Probabilities for Linear Models*

Description

This is the method add_probs uses if the model is of class lm. Probabilities are calculated parametrically, using a pivotal quantity.

Usage

```
## S3 method for class 'lm'
add_probs(
  df,
  fit,
  q,
  name = NULL,
  yhatName = "pred",
  comparison = "<",
  log_response = FALSE,
  ...
)
```

Arguments

df	A data frame of new data.
fit	An object of class lm. Predictions are made with this object.
q	A real number. A quantile of the response distribution.
name	NULL or a string. If NULL, probabilities automatically will be named by add_probs, otherwise, the probabilities will be named name in the returned data frame.
yhatName	A character vector of length one. Names of the
comparison	"<", or ">". If comparison = "<", then $Pr(Y\|x < q)$ is calculated for each observation in df. Otherwise, $Pr(Y\|x > q)$ is calculated.
log_response	A logical. Default is FALSE. Set to TRUE if the model is log-linear: $\log(Y) = X\beta + \epsilon$.
...	Additional arguments.

Value

A dataframe, df, with predicted values and probabilities attached.

See Also

add_ci.lm for confidence intervals for lm objects, add_pi.lm for prediction intervals of lm objects, and add_quantile.lm for response quantiles of lm objects.

Examples

```
# Fit a linear model
fit <- lm(dist ~ speed, data = cars)

# Calculate the probability that a new dist will be less than 20,
# given the model.
add_probs(cars, fit, q = 20)

# Calculate the probability that a new dist will be greater than
# 30, given the model.
add_probs(cars, fit, q = 30, comparison = ">")
```

---

add_probs.lmerMod *Response Probabilities for Linear Mixed Models*

Description

This function is one of the methods of add_probs, and is called automatically when add_probs is used on a fit of class lmerMod.

Usage

```
## S3 method for class 'lmerMod'
add_probs(
  df,
  fit,
  q,
  name = NULL,
  yhatName = "pred",
  comparison = "<",
  type = "parametric",
  includeRanef = TRUE,
  nSims = 10000,
  log_response = FALSE,
  ...
)
```

Arguments

df	A data frame of new data.
fit	An object of class lmerMod.
q	A real number. A quantile of the conditional response distribution.
name	NULL or a string. If NULL, probabilities automatically will be named by add_probs, otherwise, the probabilities will be named name in the returned data frame.

yhatName	A string. Name of the vector of predictions.
comparison	A character vector of length one. Must be either "<" or ">". If comparison = "<", then $Pr(Y\|x < q)$ is calculated for each x in the new data, df. Otherwise, $Pr(Y\|x > q)$ is calculated.
type	A string, either "parametric" or "boot". Determines the method used to determine the probabilities.
includeRanef	A logical. If TRUE, probabilities and predictions will be calculated at the group level. If FALSE, random effects will not be included, and probabilities will be calculated at the population level.
nSims	A positive integer. If type = "boot", nSims will determine the number of bootstrap simulations to perform.
log_response	A logical. Set to TRUE if your model is a log-linear mixed model: $log(Y) = X\beta + Z\gamma + \epsilon$.
...	Additional arguments.

Details

It is recommended that one perform a parametric bootstrap to determine these probabilities. To do so, use the option type = "boot".

Value

A dataframe, df, with predictions and probabilities attached.

See Also

add_ci.lmerMod for confidence intervals for lmerMod objects, add_pi.lmerMod for prediction intervals of lmerMod objects, and add_quantile.lmerMod for response quantiles of lmerMod objects.

Examples

```
dat <- lme4::sleepstudy

# Fit a random intercept model
fit <- lme4::lmer(Reaction ~ Days + (1|Subject), data = lme4::sleepstudy)

# What is the probability that a new reaction time will be less
# than 300? (given the random effects).
add_probs(dat, fit, q = 300)

# What is the probability that a new reaction time will be greater
# than 300? (ignoring the random effects).
add_probs(dat, fit, q = 300, includeRanef = FALSE, comparison = ">")
```

add_probs.negbin *Response Probabilities for Negative Binomial Models*

Description

This is the method add_probs uses if the model fit is an object of class negbin. Probabilities are determined through simulation, using the same method as add_pi.negbin.

Usage

```
## S3 method for class 'negbin'
add_probs(
  df,
  fit,
  q,
  name = NULL,
  yhatName = "pred",
  comparison = "<",
  nSims = 2000,
  ...
)
```

Arguments

df	A data frame of new data.
fit	An object of class negbin. Predictions are made with this object.
q	A double. A quantile of the response distribution.
name	NULL or a string. If NULL, probabilities automatically will be named by add_probs(), otherwise, the probabilities will be named name in the returned data frame.
yhatName	A string. Name of the vector of predictions.
comparison	A character vector of length one. Permitted arguments are "=", "<", "<=", ">", or ">=". The default value is "<".
nSims	A positive integer. Controls the number of simulated draws.
...	Additional arguments.

Value

A dataframe, df, with predicted values and probabilities attached.

See Also

add_ci.negbin for confidence intervals for negbin objects, add_pi.negbin for prediction intervals of negbin objects, and add_quantile.negbin for response quantiles of negbin objects.

Examples

```
x1 <- rnorm(100, mean = 1)
y <- MASS::rnegbin(n = 100, mu = exp(1 + x1), theta = 5)
df <- data.frame(x1 = x1, y = y)
fit <- MASS::glm.nb(y ~ x1, data = df)
add_probs(df, fit, q = 50)
```

`add_probs.survreg` *Confidence Intervals for the Survivor Function of Accelerated Failure Time Models*

Description

This function is one of the methods of `add_probs` and is automatically called when an object of class `survreg` is passed to `add_probs`.

Usage

```
## S3 method for class 'survreg'
add_probs(
  df,
  fit,
  q,
  name = NULL,
  yhatName = "median_pred",
  comparison = "<",
  confint = TRUE,
  alpha = 0.05,
  ...
)
```

Arguments

`df`	A data frame of new data.
`fit`	An object of class `survreg`. Predictions are made with this object.
`q`	A double. A quantile of the survival time distribution. In survival applications this is the time of event.
`name`	`NULL` or a string. If `NULL`, probabilities automatically will be named by `add_probs()`, otherwise, the probabilities will be named `name` in the returned data frame.
`yhatName`	A string. Name of the vector of predictions.
`comparison`	A character vector of length one. If `comparison = "<"`, then $Pr(Y\|X < q)$ is calculated. If `comparison = ">"`, the survivor function at time `q` is calculated.
`confint`	A logical. If `TRUE`, confidence intervals for the estimated probabilities will be calculated and appended to `df`.

alpha	A number. Control the confidence level of the confidence intervals if confint = TRUE.
...	Additional arguments.

Details

Confidence intervals may be produced for estimated probabilities of accelerated failure time models. Presently, confidence intervals may be computed for lognormal, weibull, exponential, and loglogistic failure time models. If comparison = "<", confidence intervals are made for the probability that a failure will be observed before q. Similarly, if comparison = ">", confidence intervals will be formed for the probability that a unit fails after q. In the survival literature, comparison = ">" corresponds to estimating the survivor function, *S(q)*.

Confidence intervals are produced parametrically via the Delta Method. Simulations show that under a mild to moderate amount of censoring, this method performs adequately.

The logistic transformation is applied to ensure that confidence interval bounds lie between 0 and 1.

Note: Due to a limitation, the Surv object must be specified in survreg function call. See the examples section for one way to do this.

Note: add_probs.survreg cannot inspect the convergence of fit. Poor maximum likelihood estimates will result in poor confidence intervals. Inspect any warning messages given from survreg.

Value

A dataframe, df, with predicted medians, probabilities, and confidence intervals for predicted probabilities attached.

References

For the logistic transformation of estimated probabilities and error bounds: Meeker, William Q., and Luis A. Escobar. Statistical methods for reliability data. John Wiley & Sons, 2014. (Chapter 8)

For a discussion of forming confidence intervals for survival probabilities: Harrell, Frank E. Regression modeling strategies. Springer, 2015. (Chapter 17)

See Also

add_ci.survreg for confidence intervals for survreg objects, add_pi.survreg for prediction intervals of survreg objects, and add_quantile.survreg for response quantiles of survreg objects.

Examples

```
## Define a data set.
df <- survival::stanford2
## remove a covariate with missing values.
df <- df[, 1:4]
## next, create the Surv object inside the survreg call:
fit <- survival::survreg(survival::Surv(time, status) ~ age + I(age^2),
                         data = df, dist = "lognormal")
## Calculate the level 0.75 quantile wit CIs for that quantile
```

```
add_probs(df, fit, q = 500, name = c("Fhat", "lwr", "upr"))

## Try a weibull model for the same data:
fit2 <- survival::survreg(survival::Surv(time, status) ~ age + I(age^2),
                          data = df, dist = "weibull")
## Calculate the level 0.75 quantile with CIs for the quantile
add_probs(df, fit2, q = 500, name = c("Fhat", "lwr", "upr"))
```

---

add_quantile *Add Regression Quantiles to Data Frames*

Description

This is a generic function to append regression quantiles to a data frame. A regression quantile q is a point such that $Pr(Response|Covariates < q) = p$. These quantiles are generated for the fitted value of each observation in `df`. Quantiles are then appended to `df` and returned to the user as a data frame.

Usage

```
add_quantile(df, fit, p = 0.5, name = NULL, yhatName = "pred", ...)
```

Arguments

`df`	A data frame of new data.
`fit`	An object of class `lm`, `glm`, or `lmerMod`. Predictions are made with this object.
`p`	A double. A probability that determines the quantile. Must be between 0 and 1.
`name`	`NULL` or a string. If `NULL`, quantiles automatically will be named by `add_quantile()`, otherwise, the quantiles will be named `name` in the returned data frame.
`yhatName`	A string. Name of the vector of predictions.
`...`	Additional arguments

Details

For more specific information about the arguments that are applicable for each type of model, consult:

- `add_quantile.lm` for linear regression response quantiles
- `add_quantile.glm` for generalized linear regression response quantiles
- `add_quantile.lmerMod` for linear mixed models response quantiles
- `add_quantile.glmerMod` for generalized linear mixed models response quantiles
- `add_quantile.survreg` for accelerated failure time response quantiles

Note: Except in `add_ci.survreg`, the quantiles that `add_quantile` calculates are on the distribution of $Y|x$, not $E[Y|x]$. That is, they use the same distribution that determines a prediction interval, not the distribution that determines a confidence interval.

Value

A dataframe, `df`, with predicted values and level-*p* quantiles attached.

See Also

`add_ci` for confidence intervals, `add_probs` for response level probabilities, and `add_pi` for prediction intervals

Examples

```
# Fit a linear model
fit <- lm(dist ~ speed, data = cars)

# Find the 0.4 quantile (or 40th percentile) of new distances for
# each observations in cars, conditioned on the linear model.
add_quantile(cars, fit, p = 0.4)

# Fit a Poisson model
fit2 <- glm(dist ~ speed, family = "poisson", data = cars)
# Find the 0.4 quantile (or 40th percentile) of new distances for
# each observation in cars, conditioned on the Poisson model.
add_quantile(cars, fit2, p = 0.4)

# Fit a random intercept linear mixed model
fit3 <- lme4::lmer(Reaction ~ Days + (1|Subject), data = lme4::sleepstudy)
# Find the 0.4 quantile (or 40 percentile) of reaction times for
# each observation in the sleepstudy data. Condition on the model and random effects.
add_quantile(lme4::sleepstudy, fit3, p = 0.4, type = "parametric")
```

---

add_quantile.glm *Quantiles for the Response of a Generalized Linear Model*

Description

This function is one of the methods of `add_quantile`. Currently, you can only use this function to compute the quantiles of the response of Poisson, Quasipoisson, Gamma, or Gaussian regression models. Quantile estimates for Bernoulli response variables (i.e., logistic regression) are not supported.

Usage

```
## S3 method for class 'glm'
add_quantile(df, fit, p, name = NULL, yhatName = "pred", nSims = 2000, ...)
```

Arguments

df	A data frame of new data.
fit	An object of class glm. Predictions are made with this object.
p	A real number between 0 and 1. Sets the probability level of the quantiles.
name	NULL or a string. If NULL, quantiles automatically will be named by add_quantile, otherwise, they will be named name.
yhatName	A string. Name of the vector of predictions.
nSims	A positive integer. Set the number of simulated draws to use.
...	Additional arguments.

Details

Quantiles of generalized linear models are determined by add_quantile through a simulation using arm::sim. If a Quasipoisson regression model is fit, simulation using the Negative Binomial distribution is performed, see Gelman and Hill (2007).

If add_quantile.glm is called on a Gaussian GLM with identity link function, the returned quantiles are identical to those of add_quantile.lm. If a different link function is used, the appropriate inverse transformation is applied.

Value

A dataframe, df, with predicted values and level p quantiles attached.

See Also

add_ci.glm for confidence intervals for glm objects, add_pi.glm for prediction intervals of glm objects, and add_probs.glm for response probabilities of glm objects.

Examples

```
# Fit a Poisson GLM
fit <- glm(dist ~ speed, data = cars, family = "poisson")

# What is the 0.3-quantile (or 30th percentile) of new distances,
# given the Poisson model?
add_quantile(cars, fit, p = 0.3)

# As above, but now find the 0.5-quantile (50th percentile), change
# the number of simulations to run, and give the vector of
# quantiles a custom name.
add_quantile(cars, fit, p = 0.5, name = "my_quantile", nSims = 300)
```

add_quantile.glmerMod *Response Quantiles for Generalized Linear Mixed Model Predictions*

Description

This function is one of the methods for add_pi, and is called automatically when add_pi is used on a fit of class glmerMod. This function is experimental.

Usage

```
## S3 method for class 'glmerMod'
add_quantile(
  df,
  fit,
  p,
  name = NULL,
  yhatName = "pred",
  type = "boot",
  includeRanef = TRUE,
  nSims = 10000,
  ...
)
```

Arguments

df	A data frame of new data.
fit	An object of class glmerMod.
p	A real number between 0 and 1. Sets the probability level of the quantiles.
name	NULL or a string. If NULL, quantile automatically will be named by add_quantile
yhatName	NULL or a string. Name of the predictions vector.
type	A string. Must be "boot", If type = "boot", then add_ci calls lme4::simulate to calculate the confidence intervals. This method may be time consuming, but is applicable with random slope and random intercept models.
includeRanef	A logical. Default is TRUE. Set whether the predictions and intervals should be made conditional on the random effects. If FALSE, random effects will not be included.
nSims	A positive integer. Controls the number of bootstrap replicates.
...	Additional arguments.

Details

If IncludeRanef is False, random slopes and intercepts are set to 0. Unlike in 'lmer' fits, settings random effects to 0 does not mean they are marginalized out. Consider generalized estimating equations if this is desired.

Value

A dataframe, df, with predicted values and quantiles attached.

See Also

add_pi.glmerMod for prediction intervals of glmerMod objects, add_probs.glmerMod for conditional probabilities of glmerMod objects, and add_ci.glmerMod for confidence intervals of glmerMod objects.

Examples

```
n <- 300
x <- runif(n)
f <- factor(sample(1:5, size = n, replace = TRUE))
y <- rpois(n, lambda = exp(1 - 0.05 * x * as.numeric(f) + 2 * as.numeric(f)))
df <- data.frame(x = x, f = f, y = y)
fit <- lme4::glmer(y ~ (1+x|f), data=df, family = "poisson")

add_quantile(df, fit, name = "quant0.6", p = 0.6, nSims = 500)
```

---

add_quantile.lm *Quantiles for the Response of a Linear Model*

Description

This function is one of the methods of add_quantile. It is called automatically when add_quantile is called on objects of class lm.

Usage

```
## S3 method for class 'lm'
add_quantile(
  df,
  fit,
  p,
  name = NULL,
  yhatName = "pred",
  log_response = FALSE,
  ...
)
```

Arguments

df	A data frame of new data.
fit	An object of class lm. Predictions are made with this object.
p	A real number between 0 and 1. Sets the level of the quantiles.

name	NULL or a string. If NULL, quantiles automatically will be named by add_quantile, otherwise, they will be named name.
yhatName	A string. Name of the vector of predictions.
log_response	A logical. If TRUE, quantiles will be generated for the prediction made with a log-linear model: $\log(Y) = X\beta + \epsilon$. These quantiles will be on the scale of the original response, Y.
...	Additional arguments.

Details

Quantiles for linear models are determined parametrically, by applying a pivotal quantity to the distribution of $Y|x$.

Value

A dataframe, df, with predicted values and level - p quantiles attached.

See Also

add_ci.lm for confidence intervals for lm objects, add_pi.lm for prediction intervals of lm objects, and add_probs.lm for response probabilities of lm objects.

Examples

```
# Fit a linear Model
fit <- lm(dist ~ speed, data = cars)

# Find the 0.7-quantile (70th percentile) of new distances, given
# the linear model fit.
add_quantile(cars, fit, p = 0.7)

# As above, but with a custom name for the vector of quantiles
add_quantile(cars, fit, p = 0.7, name = "my_quantile")
```

---

add_quantile.lmerMod *Quantiles for the Response of a Linear Mixed Model*

Description

This function is one of the methods for add_quantile and is called automatically when add_quantile is applied to an object of class lmerMod.

Usage

```
## S3 method for class 'lmerMod'
add_quantile(
  df,
  fit,
  p,
  name = NULL,
  yhatName = "pred",
  includeRanef = TRUE,
  type = "boot",
  nSims = 10000,
  log_response = FALSE,
  ...
)
```

Arguments

df	A data frame of new data.
fit	An object of class lm. Predictions are made with this object.
p	A real number between 0 and 1. Determines the probability level of the quantiles.
name	NULL or a string. If NULL, quantiles automatically will be named by add_quantile, otherwise, they will be named name.
yhatName	A string. Determines the name of the column of predictions.
includeRanef	The random effects be included or not? If TRUE, quantiles will be calculated at the "group level". Otherwise, quantiles will be calculated at the "population level", where random effects are set to 0.
type	A string. Options are "parametric" or "boot".
nSims	A positive integer. Set the number of bootstrap simulations to perform. Only applied when type = "boot".
log_response	A logical. Set to TRUE if the model is a log-linear mixed model: $\log(Y) = X\beta + Z\gamma + \epsilon$.
...	Additional arguments.

Details

add_qauntile.lmerMod may use one of two different methods for determining quantiles: a parametric method or a parametric bootstrap method (via lme4::simulate). The parametric method is the default. Only use the parametric method (type = "parametric") if fit is a random intercept model, e.g. fit = lmer(y ~ x + (1|group)). If your model of interest is random slope and random intercept, use the parametric bootstrap method (type = "boot").

Value

A dataframe, df, with predicted values and level-p quantiles attached.

See Also

add_ci.lmerMod for confidence intervals for lmerMod objects, add_pi.lmerMod for prediction intervals of lmerMod objects, and add_probs.lmerMod for response probabilities of lmerMod objects.

Examples

```
dat <- lme4::sleepstudy

# Fit a random intercept model
fit <- lme4::lmer(Reaction ~ Days + (1|Subject), data = lme4::sleepstudy)

# Using the parametric method: given the model fit, what value
# of reaction time do we expect half of new reaction times to fall
# under?
add_quantile(dat, fit, p = 0.5)

# Using the parametric method:
# as above, but we ignore the random effects.
add_quantile(dat, fit, p = 0.5, includeRanef = FALSE)
```

---

add_quantile.negbin *Quantiles for the Response of a Negative Binomial Regression*

Description

This function is one of the methods of add_quantile.

Usage

```
## S3 method for class 'negbin'
add_quantile(df, fit, p, name = NULL, yhatName = "pred", nSims = 2000, ...)
```

Arguments

df	A data frame of new data.
fit	An object of class negbin. Predictions are made with this object.
p	A real number between 0 and 1. Sets the probability level of the quantiles.
name	NULL or a string. If NULL, quantiles automatically will be named by add_quantile, otherwise, they will be named name.
yhatName	A string. Name of the vector of predictions.
nSims	A positive integer. Set the number of simulated draws to use.
...	Additional arguments.

Details

Quantiles of Negative Binomial linear models are determined by add_quantile through a simulation using arm::sim.

Value

A dataframe, df, with predicted values and level *p* quantiles attached.

See Also

add_ci.negbin for confidence intervals for negbin objects, add_pi.negbin for prediction intervals of negbin objects, and add_probs.negbin for response probabilities of negbin objects.

Examples

```
x1 <- rnorm(100, mean = 1)
y <- MASS::rnegbin(n = 100, mu = exp(1 + x1), theta = 5)
df <- data.frame(x1 = x1, y = y)
fit <- MASS::glm.nb(y ~ x1, data = df)
add_quantile(df, fit, p = 0.3)
```

---

add_quantile.survreg *Confidence Intervals for Predicted Survival Time Quantiles of Accelerated Failure Time Models*

Description

This function is one of the methods of add_quantile and is automatically called when an object of class survreg is passed to add_quantile.

Usage

```
## S3 method for class 'survreg'
add_quantile(
  df,
  fit,
  p = 0.5,
  name = NULL,
  yhatName = "median_pred",
  confint = TRUE,
  alpha = 0.1,
  ...
)
```

Arguments

df	A data frame of new data.
fit	An object of class survreg. Predictions are made with this object.
p	A real number between 0 and 1. Sets the probability level of the quantiles.
name	NULL or a character vector of length 3. If NULL, quantiles automatically will be named by add_quantile, otherwise, they will be named name.

yhatName	A string. Name of the vector of predictions.
confint	A logical. If TRUE, confidence intervals for the quantiles are also appended to df.
alpha	A number. Controls the confidence level of the confidence intervals if confint = TRUE.
...	Additional arguments.

Details

add_quantile.survreg produces quantiles for the estimated distribution of survival times from a survreg object. Estimated quantiles (such as the median survival time) may be calculated for a range of distributions including lognormal, exponential, weibull, and loglogistic models. add_quantile.survreg can compute quantiles through a parametric method based on the Delta Method. Generally, this method performs well under a mild to moderate amount of censoring. Parametric intervals are calculated using a transformation of the confidence intervals produced by predict.survreg and are mathematically identical to intervals calculated by a manual Delta Method.

Unlike other add_quantile methods, add_quantile.survreg additionally produces confidence intervals for predicted quantiles by default. This may optionally be disabled by switching the confint argument.

Note: Due to a limitation, the Surv object must be specified in survreg function call. See the examples section for one way to do this.

Note: add_quantile.survreg cannot inspect the convergence of fit. Poor maximum likelihood estimates will result in poor confidence intervals. Inspect any warning messages given from survreg.

Value

A dataframe, df, with predicted medians, level p quantiles, and confidence intervals attached.

References

For descriptions of the log-location scale models supported: Meeker, William Q., and Luis A. Escobar. Statistical methods for reliability data. John Wiley & Sons, 2014. (Chapter 4)

For a description of the multivariate Delta method: Meeker, William Q., and Luis A. Escobar. Statistical methods for reliability data. John Wiley & Sons, 2014. (Appendix B.2)

For a description of Delta Method Confidence Intervals: Meeker, William Q., and Luis A. Escobar. Statistical methods for reliability data. John Wiley & Sons, 2014. (Chapter 8)

See Also

add_ci.survreg for confidence intervals survreg objects, add_pi.survreg for prediction intervals of survreg objects, and add_probs.survreg for survival probabilities of survreg objects.

Examples

```
## Define a data set:
df <- survival::stanford2
## remove a covariate with missing values:
df <- df[, 1:4]
## next, create the Surv object inside the survreg call:
fit <- survival::survreg(survival::Surv(time, status) ~ age + I(age^2),
                         data = df, dist = "lognormal")
## Calculate the level 0.75 quantile wit CIs for that quantile
add_quantile(df, fit, p = 0.75, name = c("quant", "lwr", "upr"))

## Try a weibull model for the same data:
fit2 <- survival::survreg(survival::Surv(time, status) ~ age + I(age^2),
                          data = df, dist = "weibull")
## Calculate the level 0.75 quantile with CIs for the quantile
add_quantile(df, fit2, p = 0.75, name = c("quant", "lwr", "upr"))
```

Index