# Package 'glmglrt'

October 13, 2022

**Type** Package

**Title** GLRT P-Values in Generalized Linear Models

**Version** 0.2.2

**Maintainer** André GILLIBERT <andre.gillibert@chu-rouen.fr>

**Author** André GILLIBERT [aut, cre]

**Description** Provides functions to compute Generalized Likelihood Ratio Tests (GLRT) also known as Likelihood Ratio Tests (LRT) and Rao's score tests of simple and complex contrasts of Generalized Linear Models (GLMs). It provides the same interface as summary.glm(), adding GLRT P-values, less biased than Wald's P-values and consistent with profile-likelihood confidence interval generated by confint().
See Wilks (1938) <doi:10.1214/aoms/1177732360> for the LRT chi-square approximation.
See Rao (1948) <doi:10.1017/S0305004100023987> for Rao's score test.
See Wald (1943) <doi:10.2307/1990256> for Wald's test.

**License** GPL-2

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 3.5)

**Imports** stats, parameters (>= 0.1.0), MASS

**RoxygenNote** 7.1.1

**Suggests** testthat (>= 2.1.0), lme4, nlme, datasets, nnet, survival, lmerTest, mgcv, gam, multcomp

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2020-08-07 08:40:11 UTC

## R topics documented:

## `confint_contrast` *Confidence interval estimation of contrasts*

### Description

This S3 generic function allows the computation of confidence intervals of contrasts (i.e. linear combinations) of fixed-effects in many models. The default implementation computes Wald's confidence intervals with any model as long as it consistently implements `fixcoef`, `vcov_fixcoef` and `df_for_wald`. It is also specialized for GLMs with Wald's, LRT and Rao's confidence intervals and may be specialized with other models.

### Usage

```
confint_contrast(
  model,
  contrast,
  method = NULL,
  level = 0.95,
  alternative = c("two.sided", "less", "greater"),
  ...
)

## Default S3 method:
confint_contrast(
  model,
  contrast,
  method = NULL,
  level = 0.95,
  alternative = c("two.sided", "less", "greater"),
  clevel_logit_tol = 1e-05,
  deriv_rel_SE = 1e-04,
  ...,
  force = FALSE,
  debuglevel = 1
)
```

## Arguments

| | |
|---|---|
| `model` | a fitted statistical model such as a glm or a coxph. |
| `contrast` | numeric vector of the same length as the number of coefficients in the model; it describes the contrast `sum(contrast*fixcoef(model))`. |
| `method` | character string value; specification of the algorithm used (implementation dependent). NULL must be accepted. Suggested values are "LRT" for inverted likelihood ratio test, "Rao" for inverted Rao's score test, "Wald" for inverted Wald's test. |
| `level` | numeric value between 0 and 1; nominal confidence level. |
| `alternative` | character value; either "two.sided", "less" or "greater", specifying a two-sided or one-sided confidence interval. |
| `...` | Additional parameters that may be used by some implementations. |
| `clevel_logit_tol` | numeric value; the difference of logit(1-level) that can be tolerated for convergence of the algorithm. |
| `deriv_rel_SE` | numeric value; the delta for the numeric derivative, used for the Newton-Raphson algorithm applied to the logit(1-pvalue). It is expressed as a multiplicative factor for the Standard Error of the contrast. |
| `force` | logical; if TRUE, force computation of P-values in case of convergence problems. |
| `debuglevel` | integer value; set to 0 (default) to disable warnings, 1 to enable warnings and 2 to enable warnings and notes. |

## Details

This function should consistent with `estimate_contrast` and `p_value_contrast` as they are designed to be used together. If a null hypothesis (H0) is specified, it MUST be ignored by `confint_contrast` as in `estimate_contrast`. If you want to make it consistent with `p_value_contrast` you may subtract H0 from the output of `estimate_contrast` and `confint_contrast`.

When alternative is "less" or "greater", one-sided confidence intervals are generated.

## Value

A vector of length 2. The first value MUST be named "lower" and be the lower bound of the confidence interval. The second value MUST be named "upper" and be the upper bound of the confidence interval.

## Methods (by class)

- `default`: Default implementation Supports Wald's test on a wide range of models, including `lm`, `mlm`, `stats::glm`, `negbin`, `MASS::polr`, `MASS::rlm` (with normality assumptions, defeating the purpose of rlm), `nlme::lme`, `nlme::gls`, `lme4::lmer`, `lme4::glmer`, `mgcv::gam`, `gam::gam`, `survival::coxph`, `survival::survreg`, `nnet::multinom`, `stats::nls`.

  It can be easily extended by implementing three generic functions: `fixcoef`, `vcov_fixcoef` and `df_for_wald`. If the implementations of `coef`, `vcov` and `df.residual` are consistent, you do not have to implement `fixcoef`, `vcov_fixcoef` and `df_for_wald`.

It also provides method="LRT" and method="Rao" on stats::glm and method="LRT" on negative binomials (negbin) models. It is implemented by inverting tests performed by p_value_contrast, with the help of the Newton-Raphson algorithm on the logit of the two-sided P-value.

## See Also

Other Contrast functions: estimate_confint_contrast(), estimate_contrast(), p_value_contrast()

## Examples

```
data(mtcars)
model1 = glm(family="gaussian", data=mtcars, hp ~ 0+factor(gear))
# do cars with 5 gears have more horse power (hp) than cars with 4 gears ?
confint_contrast(model1, c(0,-1,1))

# now, we fit an equivalent model (same distribution and same predictions)
model2 = glm(family=gaussian(log), data=mtcars, hp ~ 0+factor(gear))

# do cars with 5 gears have at least twice the horse power than cars with 4 gears ?

confint_contrast(model2, c(0,-1,0.5))
```

---

df_for_wald *Gets the degree of freedom for Wald tests involving the model*

---

## Description

This generic function is used by p_value_contrast.default to get the number of degrees of freedom of the t distribution that approximates the point estimate of the contrast divided by its standard error.

## Usage

```
df_for_wald(object, ...)

## S3 method for class 'glm'
df_for_wald(object, ...)

## Default S3 method:
df_for_wald(object, ...)
```

## Arguments

| | |
|---|---|
| object | statistical model; |
| ... | Unused by p_value_contrast.default, but may be useful to some custom specializations. |

### Details

This function is quite similar to `df.residual` but it should return Inf when the Student's t distribution is less appropriate than the normal distribution.

### Value

A finite value or Inf for normal distribution approximation.

### Methods (by class)

- `glm`: Returns `df.residual` for linear gaussian models and Inf for all other models in order to make Wald's tests consistent with the behavior of `stats::summary.glm(object)`
- `default`: Simple proxy to `df.residual` but replaces NAs with Inf

### See Also

Other Wald-related functions: `fixcoef()`, `p_value_contrast()`, `vcov_fixcoef()`

### Examples

```
# 10 observations, one coefficient, 9 degrees of freedom
df_for_wald(glm(I(1:10) ~ 1))
# returns Inf (non-gaussian-identity model)
df_for_wald(glm(family="poisson", c(10,20,30) ~ 1))
data(mtcars)
# returns Inf (non-gaussian-identity model)
df_for_wald(glm(family="binomial", data=mtcars, I(hp > median(hp)) ~ cyl))
```

---

## `estimate_confint_contrast`

*Computes point estimates, confidence intervals and P-values of a contrast*

---

### Description

This function combines outputs from `estimate_contrast`, `confint_contrast` and `p_value_contrast` to provide a 4-values vector with point estimate (1st value), lower and upper boundaries of the confidence interval (2nd and 3rd values) and P-value (4th value) comparing the contrast to `H0`.

### Usage

```
estimate_confint_contrast(
  model,
  contrast,
  method = NULL,
  level = 0.95,
  force = FALSE,
  debuglevel = 1,
  H0 = 0,
  alternative = c("two.sided", "less", "greater"),
  ...
)
```

## Arguments

| | |
|---|---|
| `model` | a fitted statistical model such as a glm or a coxph. |
| `contrast` | numeric vector of the same length as the number of coefficients in the model; it describes the contrast `sum(contrast*fixcoef(model))`. |
| `method` | character string value; specification of the algorithm used (implementation dependent). NULL must be accepted. The default method is "Wald". With the default `confint_contrast` and `p_value_contrast`, the only supported values are: "Wald", "wald" and "SlowWald". All three are equivalent but "SlowWald" is slower and is used for debug purpose only. If `confint_contrast` and `p_value_contrast` are specialized, they may provide other methods. |
| `level` | numeric value between 0 and 1; nominal two-sided confidence level of the confidence interval. |
| `force` | logical; if TRUE, force computation of P-values in case of convergence problems. |
| `debuglevel` | integer value; set to 0 (default) to disable warnings, 1 to enable warnings and 2 to enable warnings and notes. |
| `H0` | numeric value; the value of the contrast under the null hypothesis. |
| `alternative` | a character string specifying the alternative hypothesis, |
| `...` | Additional parameters that may be used by some implementations. |

## Details

When alternative is "less" or "greater", a one-sided confidence interval and a one-sided P-value are generated. If H0 is not zero, the P-value compares the estimate to the value of H0, but the estimate and confidence interval are unchanged.

## See Also

Other Contrast functions: `confint_contrast()`, `estimate_contrast()`, `p_value_contrast()`

## Examples

```
data(mtcars)
model1 = glm(family="gaussian", data=mtcars, hp ~ 0+factor(gear))
# do cars with 5 gears have more horse power (hp) than cars with 4 gears ?
estimate_confint_contrast(model1, c(0,-1,1))

# now, we fit an equivalent model (same distribution and same predictions)
model2 = glm(family=gaussian(log), data=mtcars, hp ~ 0+factor(gear))

# do cars with 5 gears have at least twice the horse power than cars with 4 gears ?

estimate_confint_contrast(model2, c(0,-1,0.5))
```

## `estimate_contrast` *Point estimates of contrasts*

### Description

This S3 generic function allows the computation of point estimates of contrasts (i.e. linear combinations) of fixed-effects in many models The default implementation computes Wald's confidence intervals with any model as long as it implements `fixcoef`, returning a vector of fixed effects.

### Usage

```
estimate_contrast(model, contrast, method = NULL, ...)

## Default S3 method:
estimate_contrast(model, contrast, method = NULL, ...)
```

### Arguments

| | |
|---|---|
| `model` | a fitted statistical model such as a glm or a coxph. |
| `contrast` | numeric vector of the same length as the number of coefficients in the model; it describes the contrast `sum(contrast*fixcoef(model))`. |
| `method` | character string value; specification of the algorithm used (implementation dependent). NULL must be accepted. Suggested values are "ML" for maximum-likelihood, "REML" for restricted maximum-likelihood and "OLS" for ordinary least squares. |
| `...` | Additional parameters that may be used by some implementations. |

### Details

This function should consistent with `confint_contrast` and `p_value_contrast` as they are designed to be used together. If a null hypothesis (H0) is specified, it MUST be ignored by `estimate_contrast`. If you want to make it consistent with p_value_contrast you may substract H0 from the output of `estimate_contrast` and `confint_contrast`.

### Value

A single numeric value (vector of length 1) equal to the point estimate of the contrast, with the name "pvalue".

### Methods (by class)

- `default`: Compute contrasts of fixed-effects in any model implementing `fixcoef`. It basically computes `sum(fixcoef(model) * contrast)`.

### See Also

Other Contrast functions: `confint_contrast()`, `estimate_confint_contrast()`, `p_value_contrast()`

## Examples

```
data(mtcars)
model1 = glm(family="gaussian", data=mtcars, hp ~ 0+factor(gear))
# do cars with 5 gears have more horse power (hp) than cars with 4 gears ?
estimate_contrast(model1, c(0,-1,1))

# now, we fit an equivalent model (same distribution and same predictions)
model2 = glm(family=gaussian(log), data=mtcars, hp ~ 0+factor(gear))

# do cars with 5 gears have at least twice the horse power than cars with 4 gears ?

estimate_contrast(model1, c(0,-1,0.5))
```

---

## `fixcoef` *Generic function to get fixed effects of a model*

---

### Description

This is a generic S3 function that gets point estimates of fixed effects of a statistical model, implemented on a wide range of models and that can be extended to new models.

### Usage

```
fixcoef(model, ...)

## S3 method for class 'lmerMod'
fixcoef(model, ...)

## S3 method for class 'glmerMod'
fixcoef(model, ...)

## S3 method for class 'lmerModLmerTest'
fixcoef(model, ...)

## S3 method for class 'lme'
fixcoef(model, ...)

## S3 method for class 'multinom'
fixcoef(model, ...)

## S3 method for class 'mlm'
fixcoef(model, ...)

## Default S3 method:
fixcoef(model, ...)
```

## Arguments

| | |
|---|---|
| `model` | a fitted statistical model |
| `...` | argument unused by `p_value_contrast.default` but that may be useful to some specializations. |

## Details

It must return only estimates of fixed-effects of a model. Random effects are ignored. The `names` of the element of this vector must be consistent with the `rownames` and `colnames` of the variance-covariance matrix that `vcov_fixcoef` returns. The `vcov_fixcoef` function, on the same model, must return a matrix with the same number and names of rows and columns as the length of the vector returned by `fixcoef`.

The functions `vcov_fixcoef` and `fixcoef` would be pointless if the behavior of `vcov` and `coef` were not inconsistent from package to package.

`fixcoef` and `vcov_fixcoef`, together with `df_for_wald` are used by `p_value_contrast.default`

## Value

Simple numeric vector with one item for each fixed effect of the model.

## Methods (by class)

- `lmerMod`: implementation for `lme4::lmer`
- `glmerMod`: implementation for `lme4::glmer`
- `lmerModLmerTest`: implementation for `lmerTest::lmer`
- `lme`: implementation for `nlme::lme`
- `multinom`: implementation for `nnet::multinom`
- `mlm`: implementation for multiple responses linear models generated by `stats::lm` when the response is a matrix. It transforms the matrix to a vector, consistent with `stats::vcov`.
- `default`: default implementation, simply calls coef(model).

## See Also

Other Wald-related functions: `df_for_wald()`, `p_value_contrast()`, `vcov_fixcoef()`

## Examples

```
data(mtcars)
fixcoef(lm(data=mtcars, hp ~ 1)) # get mean horse power of cars listed in mtcars
```

## glmglrt *glmglrt: GLRT P-Values in Generalized Linear Models*

### Description

This package has been developed to provide Generalized Likelihood Ratio Tests (GLRT) also known as Likelihood Ratio Tests (LRT) to Generalized Linear Models (GLMs). The stats package do support LRT P-values with `anova` and derived confidence intervals with `confint()`, but provides Wald's P-values with the `summary` function. This is unfortunate for two reasons: Wald's P-values may be inconsistent with profile-likelihood confidence intervals and Wald's P-values, on small samples are more biased than LRT P-values, for non-gaussian models. The `anova` function is not as simple as `summary`, since it requires manually fitting two models.

### Summary function

This package provides a way to override (see override_summary) the standard `summary.glm` function by a `summarylr` function that provides LRT and/or Rao's score P-values.

### Functions to estimate contrasts

It also provides functions `estimate_contrast`, `confint_contrast` and `p_value_contrast` to estimate contrasts of coefficients of GLMs with LRT, Rao's and Wald's hypothesis tests and confidence intervals This is an alternative to `multcomp::glht` without Wald's approximation ! It also provides a less powerful `p_value.glm` method for the S3 generic parameters::p_value. It also extends this S3 generic for a variety of models as `p_value.default`. That time, the only method supported for all models, is Wald's method.

## override_summary *Overrides the Generalized Linear Models summary methods*

### Description

This function overrides the `summary.glm` and `summary.negbin` S3 methods by the `summarylr` function in the calling environment.

### Usage

```
override_summary()
```

### Details

Although some minor compatibility issues may exist when calling this function in the global environment, most scripts should work with it. Indeed `summarylr` behaves like `summary.glm` but adds a $extra field containing P-value info. The first letter of the field name ('e') is unique, avoiding problems with scripts that access fields with short names (e.g. model$x for model$xlevels).

### See Also

Other Extended GLM summary functions: `print.summary.glmglrt()`, `summarylr()`

### Examples

```
model = glm(family="binomial", cbind(50,30) ~ 1)
override_summary()
summary(model) # Additional 'LRT P-value' column
```

---

## `print.summary.glmglrt` *Prints the summary generated by* `summarylr`

---

### Description

This function prints a `summary.glmglrt` object generated by `summarylr`. It works like the standard `summary.glm` function but additionnally displays columns showing Rao or LRT P-values.

### Usage

```
## S3 method for class 'summary.glmglrt'
print(
  x,
  ...,
  has.Pvalue = TRUE,
  tst.ind = 3,
  debuglevel = NULL,
  keep.wald = NULL
)
```

### Arguments

| | |
|---|---|
| `x` | a `summary.glmglrt` object generated by `summarylr`. |
| `...` | additional arguments passed to `stats::print.summary.glm` then `printCoefmat`. The most useful ones are `digits` and `signif.stars`. |
| `has.Pvalue` | logical value; passed to `printCoefmat`; if TRUE, the P-value column is formatted by `format.pval`. |
| `tst.ind` | integer vector of length>=0; passed to `printCoefmat`; it changes the format of these columns, assuming they are statistics columns. |
| `debuglevel` | NULL or integer value; set to NULL to use the debuglevel argument that was specified in `summarylr`, 0 (default) to disable warnings, 1 to enable warnings and 2 to enable warnings and notes. |
| `keep.wald` | NULL or logical; set to NULL to use the keep.wald argument that was specified in `summarylr`. If TRUE, the standard Wald's P-values are displayed. If FALSE, the standard Wald's P-values are hidden. |

### See Also

Other Extended GLM summary functions: `override_summary()`, `summarylr()`

### Examples

```
model = glm(family="binomial", cbind(50,30) ~ 1)
print(summarylr(model),signif.stars=FALSE,digits=10)
```

---

## ps_newtonRaphson *Newton-Raphson algorithm when parameters may not be estimable outside of their parameter space*

---

### Description

This is used by `confint_contrast.default` to find the lower and upper boundaries of the confidence interval so that they have a P-value equal to 1-level. This makes it possible to compute contrast on log-binomial models when solutions are close to the boundaries of the parameter space and Newton-Raphson may compute some intermediate values outside of the valid parameter space, leading to P-values equal to NA.

### Usage

```
ps_newtonRaphson(
  rootfun,
  x,
  xabstol = 1e-05,
  derivdelta = xabstol,
  yabstol = 1e-05,
  niter = 100,
  extraiter = 0
)
```

### Arguments

| | |
|---|---|
| `rootfun` | a function taking a single numeric value. The algorithm searches for a root (zero) of this function. |
| `x` | a single numeric value; starting value, not too far from the root of rootfun. |
| `xabstol` | a single numeric value; when two consecutive iterations of the algorithm lead to x solutions with a difference larger than xabstol, the algorithm has not yet converged and continues to run. Set to Inf, if you want to only want to rely on yabstol. |
| `derivdelta` | a single numeric value; the numeric delta used for numeric derivation of rootfun (assessed at x-derivdelta and x+derivdelta) |
| `yabstol` | a single numeric value; when the algorithm leads to a y=rootfun(x) larger (in absolute value) than yabstol, the algorithm has not yet converged and continues to run. Set to Inf, if you want to only want to rely on xabstol. |

| | |
|---|---|
| `niter` | a single integer value; the maximum number of iterations before considering that the algorithm failed to converge. |
| `extraiter` | a single integer value; the number of extra iterations performed once convergence has been found. This is useful to make sure that convergence is good on platforms having high floating point precision allowing to use, at the same time, a weak convergence tolerance to make sure it converges on platforms having low floating point precision. |

## Details

It assumes that `rootfun` is NA above the upper boundary of the parameter space and below the lower boundary of the parameter space. The parameter space must not have "holes" (i.e. valid, then invalid, then valid again). Invalid values are described by NA. The actual interval of valid values are automatically found by the algorithm, so you do not have to explicitly specify the minimum and maximum values of the parameter space.

## Value

a list object with the following fields:

`root` the x value so that rootfun(x) is as close as possible to 0

`f.root` rootfun(root)

`yabstol` the parameter of the same name

`iter` the number of iterations of the Newton-Raphson algorithm performed before reaching convergence

`estim.prec` the latest move (on the x variable) done before convergence

`prevx` the previous x value, in the algorithm, just before x=root was reached

`prevy` rootfun(prevx)

`prevder` numeric derivative of rootfun at prevx

---

`p_value.glm` *Computing p-values of hypothesis tests on coefficients of Generalized Linear Models and other*

---

## Description

This S3 method is a specialization of `parameters::p_value` for `stats::glm` models. By default, it computes Wald's P-values that are known to be more biased than LRT P-values, but the behavior can be overriden by the method="LRT" argument. This is for compatibility with the default method of `parameters::p_value`.

## Usage

```
## S3 method for class 'glm'
p_value(
  model,
  method = NULL,
  parm = NULL,
  alternative = c("two.sided", "less", "greater"),
  H0 = 0,
  debuglevel = 1,
  force = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| `model` | glm object; as obtained by calling `stats::glm` or `MASS::glm.nb`. |
| `method` | character value; may either be "LRT" (synonym "Chisq"), "Rao", "wald" (default value, synonym "Wald" and "F"). |
| `parm` | integer or character vector or NULL; specify coefficients to test, by name or indexes. the default parm=NULL outputs all coefficients. |
| `alternative` | a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter. |
| `H0` | numeric vector of length 1 or of the same length as parm; the value of the coefficient under the null hypothesis. Zero by default. |
| `debuglevel` | integer value; set to 0 (default) to disable warnings, 1 to enable warnings and 2 to enable warnings and notes. |
| `force` | logical; if TRUE, force computation of P-values in case of convergence problems. |
| `...` | Ignored arguments. Allows compatibility with the generic `parameters::p_value`. |

## Value

a data.frame with two columns; the first column, Parameter represents the name of the coefficient and p (second column) represents the P-value of the hypothesis test against H0

## Examples

```
require("parameters")
mod = glm(family="poisson", c(2,30) ~ c(0,1), offset=log(c(8,30)))
# Wald's tests (biased)
p_value(mod)
# Rao score tests (biased)
p_value(mod, method="Rao")
# LRT tests (less biased)
p_value(mod, method="LRT")

# only test slope (faster since only one test is performed)
```

```
p_value(mod, method="LRT", parm=2)
# is slope greater than log(2) ?
p_value(mod, method="LRT", parm=2, H0=log(2), alternative="greater")
```

---

`p_value_contrast` *Hypothesis tests on contrasts*

---

## Description

This S3 generic function allows the computation of P-values associated to hypothesis tests of contrasts (i.e. linear combinations) of fixed-effects in a model. The default implementation computes Wald's P-values with any model as long as it consistently implements `fixcoef`, `vcov_fixcoef` and `df_for_wald`. It is also specialized for GLMs and negative binomial models (see `MASS::glm.nb`) with Wald's, LRT and Rao's P-values and may be specialized with other models.

## Usage

```
p_value_contrast(
  model,
  contrast,
  alternative = c("two.sided", "less", "greater"),
  H0 = 0,
  method = NULL,
  ...
)

## S3 method for class 'glm'
p_value_contrast(
  model,
  contrast,
  alternative = c("two.sided", "less", "greater"),
  H0 = 0,
  method = c("LRT", "Rao", "Chisq", "F", "Wald", "wald"),
  ...,
  debuglevel = 1,
  force = FALSE
)

## Default S3 method:
p_value_contrast(
  model,
  contrast,
  alternative = c("two.sided", "less", "greater"),
  H0 = 0,
  method = "Wald",
  ...,
  debuglevel = 0,
  force = FALSE
)
```

## Arguments

| | |
|---|---|
| `model` | a fitted statistical model such as a glm or a coxph. |
| `contrast` | numeric vector of the same length as the number of coefficients in the model; it describes the contrast `sum(contrast*fixcoef(model))`. |
| `alternative` | a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter. |
| `H0` | numeric value; the value of the contrast under the null hypothesis. |
| `method` | character string value; specification of the algorithm used (implementation dependent). Suggested values are "Wald", "LRT", "Rao" and "exact" for, respectively, Wald's asymptotic normality and/or student test, the Generalized Likelihood Ratio Test, Rao's score test and non-asymptotic exact tests. Other values may be allowed. |
| `...` | Additional parameters that may be used by some implementations. |
| `debuglevel` | integer value; set to 0 (default) to disable warnings, 1 to enable warnings and 2 to enable warnings and notes. |
| `force` | logical; if TRUE, force computation of P-values in case of convergence problems. |

## Details

Every implementation MUST support specification of the alternative hypothesis (alternative argument) and null hypothesis (H0 argument).

## Value

A single numeric value (vector of length 1) equal to the one-sided (for alternative="less" or "greater") or two-sided P-value

## Methods (by class)

- `glm`: It supports Wald (method="Wald"), Generalized Likelihood Ratio Tests (method="LRT") and Rao's score tests (method="Rao"). It works for `stats::glm` models and negative binomial models (`MASS::glm.nb`) with method="LRT" and method="Wald".
- `default`: Supports Wald's test on a wide range of models, including `lm`, `mlm`, `stats::glm`, `negbin`, `MASS::polr`, `MASS::rlm` (with normality assumptions, defeating the purpose of rlm), `nlme::lme`, `nlme::gls`, `lme4::lmer`, `lme4::glmer`, `mgcv::gam`, `gam::gam`, `survival::coxph`, `survival::survreg`, `nnet::multinom`, `stats::nls`.

  It can be easily extended by implementing three generic functions: `fixcoef`, `vcov_fixcoef` and `df_for_wald`. If the implementations of `coef`, `vcov` and `df.residual` are consistent, you do not have to implement `fixcoef`, `vcov_fixcoef` and `df_for_wald`.

## See Also

Other Wald-related functions: `df_for_wald()`, `fixcoef()`, `vcov_fixcoef()`

Other Contrast functions: `confint_contrast()`, `estimate_confint_contrast()`, `estimate_contrast()`

## Examples

```
data(mtcars)
model1 = glm(family="gaussian", data=mtcars, hp ~ 0+factor(gear))
# do cars with 5 gears have more horse power (hp) than cars with 4 gears ?
p_value_contrast(model1, c(0,-1,1), alternative="greater")

# now, we fit an equivalent model (same distribution and same predictions)
model2 = glm(family=gaussian(log), data=mtcars, hp ~ 0+factor(gear))

# do cars with 5 gears have at least twice the horse power than cars with 4 gears ?

# the following two tests are equivalent
p_value_contrast(model1, c(0,-1,0.5), alternative="greater", method="LRT", H0=0)
p_value_contrast(model2, c(0,-1,1), alternative="greater", method="LRT", H0=log(2))

# the following two tests are close but not equivalent
p_value_contrast(model1, c(0,-1,0.5), alternative="greater", method="Wald", H0=0)
p_value_contrast(model2, c(0,-1,1), alternative="greater", method="Wald", H0=log(2))
```

---

`summarylr` *Summarizes a glm, adding a column of GLRT or Rao score P-values*

---

## Description

`summarylr` is an improved summary function for standard glm (stats package) adding LRT or Rao score P-values

## Usage

```
summarylr(
  object,
  dispersion = NULL,
  correlation = FALSE,
  symbolic.cor = FALSE,
  ...,
  force = FALSE,
  debuglevel = level_warning,
  method = "LRT",
  keep.wald = FALSE
)
```

## Arguments

| | |
|---|---|
| `object` | glm object; as obtained by calling `stats::glm` or `MASS::glm.nb`. |
| `dispersion` | the dispersion parameter for the family used. Either a single numerical value or NULL (the default), when it is inferred from object (see `stats::summary.glm`). |
| `correlation` | logical; if TRUE, the correlation matrix of the estimated parameters is returned and printed (see `stats::summary.glm`). |
| `symbolic.cor` | logical; if TRUE, print the correlations in a symbolic form (see `symnum`) rather than as numbers (see `stats::summary.glm`). |
| `...` | Additional arguments to be passed to `stats::summary.glm` |
| `force` | logical; if TRUE, force computation of P-values in case of convergence problems. |
| `debuglevel` | integer value; set to 0 (default) to disable warnings, 1 to enable warnings and 2 to enable warnings and notes. |
| `method` | NULL or character vector of length 0, 1 or 2; may be code"LRT" or `"Rao"` or `c("LRT", "Rao")` to compute specified P-values. You can set `method=NULL` to compute no additional P-values. |
| `keep.wald` | logical; if TRUE, the standard Wald's P-values are kept in the output of `print.summary.glmglrt`. Even if keep.wald=FALSE, the standard wald P-values are not erased from the `summary.glmglrt` object. They are only hidden by `print.summary.glmglrt`. |

## Details

This function works the same as the standard `summary.glm` function but provides additionnal parameters The core parameter `method="LRT"` makes `summarylr` adds a column `LRT P-value` to the output. This P-value is computed by repeatdly fitting the model dropping one coefficient at a time and using the `anova.glm(test="Chisq")` function to perform generalized likelihood ratio test by approximation of the deviance difference to a chi-square distribution. This provides P-values less biased than the standard Wald P-values that `summary` provides. Moreover, this LRT method is consistent with the profile likelihood confidence intervals that `confint.glm` provides. The option `method="Rao"` generates Rao's score P-values. `method="Chisq"` is synonymous to `method="LRT"`. For exhaustivity, the option `method="Wald"` (synonym "wald", "F") generates Wald's P-values. Several methods can be used, e.g. `method=c("LRT","Rao")` computes both LRT and Rao P-values. New methods may be added in the future.

Extra parameters are passed-through to the `summary.glm` function.

## Value

It returns a summary object of type `summary.glmglrt` that gets pretty printed by `link[glmglrt:print.summary.glmglrt]{` The return value is an S3 object compatible with `stats::summary.glm` but with an additional field `$extra` field having sub-fields. `$extra$pvalues` is a numeric matrix with columns "LRT P-value" and/or "Rao P-value", containing the relevant P-values. As new columns may be added in future, you should rely on column names rather than column indexes. Only P-values of methods requested in the `method` parameter are stored in this matrix. `$extra$debuglevel` is equal to the `debuglevel` passed to `summarylr`. `$extra$keep.wald` is equal to the `keep.wald` passed to `summarylr`. In case of convergence problems, the field `$extra$problem_of_convergence` will be added. It will be a character string with the value `"general"` (because model$converged = FALSE), `"all"` (because all coefficients have huge variances) or "specific" (because at least one coefficient has a huge variance). Other problem strings may be added in the future. If weights are specified in a way that make P-values invalid, the field `$extra$problem_weights` will be added as a character string describing the problem. Actually, the only known problem is "non-constant".

## See Also

Other Extended GLM summary functions: `override_summary()`, `print.summary.glmglrt()`

## Examples

```
summarylr(glm(family="binomial", cbind(5,3)~1))
data(mtcars)
# do not properly converge (warnings)
mtcars$outcome = mtcars$disp > median(mtcars$disp)
mod=glm(family=binomial(log), data=mtcars,outcome ~ 0+qsec+wt,start=c(-0.1,0.3))
summarylr(mod) # warns that P-values are not computed because model did not converge
summarylr(mod, force=TRUE) # compute P-values anyway !
# also works with negative binomial models
summarylr(MASS::glm.nb(data=mtcars, I(cyl*gear) ~ 1+wt,link="sqrt"),test="LRT")
```

---

# vcov_fixcoef *Gets the variance-covariance matrix of fixed effects of a fitted model*

---

## Description

This is a generic S3 function that gets the variance-covariance matrix of fixed effects of a statistical model, implemented on a wide range of models and that can be extended to new models.

## Usage

```
vcov_fixcoef(model, ...)

## Default S3 method:
vcov_fixcoef(model, ...)

## S3 method for class 'survreg'
vcov_fixcoef(model, ...)
```

## Arguments

| | |
|---|---|
| `model` | a fitted statistical model |
| `...` | argument unused by `p_value_contrast.default` but that may be useful to some specializations. |

## Details

It must return variance-covariance for fixed effects of a model, not random effects nor scale parameters. The `rownames` and `colnames` of the returned matrix must be consistent with `names` of `fixcoef(object)`.

The functions `vcov_fixcoef` and `fixcoef` would be pointless if the behavior of `vcov` and `coef` were not inconsistent from package to package.

`fixcoef` and `vcov_fixcoef`, together with `df_for_wald` are used by `p_value_contrast.default`

## Methods (by class)

- `default`: default implementation, simple proxy of `vcov(model)`
- `survreg`: implementation for survreg, removing the extra column for Scale

## See Also

Other Wald-related functions: `df_for_wald()`, `fixcoef()`, `p_value_contrast()`

## Examples

```
data(mtcars)
mod = lm(data=mtcars, hp ~ cyl+wt)
est = fixcoef(mod) # get estimates
SE = sqrt(diag(vcov_fixcoef(mod))) # get standard errors of estimates
z  = est/SE # get z-score of estimates
df = df_for_wald(mod) # degrees of freedom
pvalues = 2*pt(-abs(z), df=df) # get two-sided P-values
```

# Index