Package 'performance'

June 4, 2019

Type Package

Title Assessment of Regression Models Performance

Version 0.2.0

Date 2019-06-04

Maintainer Daniel Lüdecke <d.luedecke@uke.de>

Description Utilities for computing measures to assess model quality, which are not directly provided by R's 'base' or 'stats' packages. These include e.g. measures like r-squared, intraclass correlation coefficient (Nakagawa, Johnson & Schielzeth (2017) <doi:10.1098/rsif.2017.0213>), root mean squared error or functions to check models for overdispersion, singularity or zero-inflation and more. Functions apply to a large variety of regression models, including generalized linear models, mixed effects models and Bayesian models.

URL https://easystats.github.io/performance/

BugReports https://github.com/easystats/performance/issues

Depends R (>= 3.0)

Imports insight, bayestestR

Suggests AER, betareg, brms, covr, glmmTMB, lme4, loo, Matrix, MASS, mlogit, nlme, ordinal, pscl, psych, randomForest, rmarkdown, rstanarm, rstantools, see, survival, testthat

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Author Daniel Lüdecke [aut, cre] (<https://orcid.org/0000-0002-8895-3206>), Dominique Makowski [aut, ctb] (<https://orcid.org/0000-0001-5375-9967>), Philip Waggoner [aut, ctb] (<https://orcid.org/0000-0002-7825-7573>)

Repository CRAN

Date/Publication 2019-06-04 09:00:03 UTC

R topics documented:

`binned_residuals` *Binned residuals for logistic regression*

Description

Check model quality of logistic regression models.

Usage

```
binned_residuals(model, term = NULL, n_nins = NULL)
```

Arguments

`model`	A `glm`-object with binomial-family.
`term`	Name of independent variable from `x`. If not `NULL`, average residuals for the categories of `term` are plotted; else, average residuals for the estimated probabilities of the response are plotted.
`n_nins`	Numeric, the number of bins to divide the data. If `n_nins = NULL`, the square root of the number of observations is taken.

Details

Binned residual plots are achieved by "dividing the data into categories (bins) based on their fitted values, and then plotting the average residual versus the average fitted value for each bin." *(Gelman, Hill 2007: 97)*. If the model were true, one would expect about 95% of the residuals to fall inside the error bounds.

If `term` is not `NULL`, one can compare the residuals in relation to a specific model predictor. This may be helpful to check if a term would fit better when transformed, e.g. a rising and falling pattern of residuals along the x-axis (the pattern is indicated by a green line) is a signal to consider taking the logarithm of the predictor (cf. Gelman and Hill 2007, pp. 97ff).

Value

A data frame representing the data that is mapped to the plot, which is automatically plotted. In case all residuals are inside the error bounds, points are black. If some of the residuals are outside the error bounds (indicates by the grey-shaded area), blue points indicate residuals that are OK, while red points indicate model under- or overfitting for the related range of estimated probabilities.

References

Gelman, A., & Hill, J. (2007). Data analysis using regression and multilevel/hierarchical models. Cambridge; New York: Cambridge University Press.

Examples

```
model <- glm(vs ~ wt + mpg, data = mtcars, family = "binomial")
binned_residuals(model)
```

`check_autocorrelation` *Check model for independence of residuals.*

Description

Check model for independence of residuals, i.e. for autocorrelation of error terms.

Usage

```
check_autocorrelation(x, ...)

## Default S3 method:
check_autocorrelation(x, nsim = 1000, ...)
```

Arguments

`x`	A model object.
`...`	Currently not used.
`nsim`	Number of simulations for the Durbin-Watson-Test.

Details

Performs a Durbin-Watson-Test to check for autocorrelated residuals. In case of autocorrelation, robust standard errors return more accurate results for the estimates, or maybe a mixed model with error term for the cluster groups should be used.

Value

Invisibly returns the p-value of the test statistics. A p-value < 0.05 indicates autocorrelated residuals.

Examples

```
m <- lm(mpg ~ wt + cyl + gear + disp, data = mtcars)
check_autocorrelation(m)
```

check_collinearity *Check for multicollinearity of model predictors*

Description

check_collinearity() checks regression models for multicollinearity by calculating the variance inflation factor (VIF).

Usage

```
check_collinearity(x, ...)

## S3 method for class 'glmmTMB'
check_collinearity(x, component = c("all",
  "conditional", "count", "zi", "zero_inflated"), ...)
```

Arguments

x	A model object (that should at least respond to vcov(), and if possible, also to model.matrix() - however, it also should work without model.matrix()).
...	Currently not used.
component	For models with zero-inflation component, multicollinearity can be checked for the conditional model (count component, component = "conditional" or component = "count"), zero-inflation component (component = "zero_inflated" or component = "zi") or both components (component = "all"). Following model-classes are currently supported: hurdle, zeroinfl, zerocount, MixMod and glmmTMB.

Details

The variance inflation factor is a measure to analyze the magnitude of multicollinearity of model predictors. A VIF less than 5 indicates a low correlation of that predictor with other predictors. A value between 5 and 10 indicates a moderate correlation, while VIF values larger than 10 are a sign for high, not tolerable correlation of model predictors. The *Increased SE* column in the output indicates how much larger the standard error is due to the correlation with other predictors.

Value

A data frame with three columns: The name of the model term, the variance inflation factor and the factor by which the standard error is increased due to possible correlation with other predictors.

References

James, G., Witten, D., Hastie, T., & Tibshirani, R. (Hrsg.). (2013). An introduction to statistical learning: with applications in R. New York: Springer.

Examples

```
m <- lm(mpg ~ wt + cyl + gear + disp, data = mtcars)
check_collinearity(m)
```

---

check_convergence *Convergence test for mixed effects models*

Description

check_convergence() provides an alternative convergence test for merMod-objects.

Usage

```
check_convergence(x, tolerance = 0.001)
```

Arguments

x	A merMod-object.
tolerance	Indicates up to which value the convergence result is accepted. The smaller tolerance is, the stricter the test will be.

Details

check_convergence() provides an alternative convergence test for merMod-objects, as discussed here and suggested by Ben Bolker in this comment. Further details can be found in convergence.

Value

TRUE if convergence is fine and FALSE if convergence is suspicious. Additionally, the convergence value is returned as attribute.

Examples

```
library(lme4)
data(cbpp)
set.seed(1)
cbpp$x <- rnorm(nrow(cbpp))
cbpp$x2 <- runif(nrow(cbpp))

model <- glmer(
  cbind(incidence, size - incidence) ~ period + x + x2 + (1 + x | herd),
  data = cbpp,
  family = binomial()
)

check_convergence(model)
```

check_distribution *Classify the distribution of a model-family using machine learning*

Description

Choosing the right distributional family for regression models is essential to get more accurate estimates and standard errors. This function may help to check a models' distributional family and see if the model-family probably should be reconsidered. Since it is difficult to exactly predict the correct model family, consider this function as somewhat experimental.

Usage

```
check_distribution(model)
```

Arguments

model	A model (that should response to residuals()).

Details

This function uses an internal random forest model to classify the distribution from a model-family.

Note

This function is somewhat experimental and might be improved in future releases. The final decision on the model-family should also be based on theoretical aspects and other information about the data and the model.

Examples

```
library(lme4)
model <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
check_distribution(model)
```

check_heteroscedasticity *Check model for (non-)constant error variance*

Description

Check model for (non-)constant error variance.

Usage

```
check_heteroscedasticity(x, ...)
```

Arguments

x	A model object.
...	Currently not used.

Value

Invisibly returns the p-value of the test statistics. A p-value < 0.05 indicates a non-constant variance (heteroskedasticity).

Examples

```
m <- lm(mpg ~ wt + cyl + gear + disp, data = mtcars)
check_heteroscedasticity(m)
```

---

check_model *Visual check of model assumptions*

Description

Visual check of model various assumptions (normality of residuals, normality of random effects, heteroscedasticity, homogeneity of variance, multicollinearity).

Usage

```
check_model(x, ...)

## Default S3 method:
check_model(x, dot_size = 2, line_size = 0.8,
  panel = TRUE, ...)
```

Arguments

x	A model object.
...	Currently not used.
dot_size	Size of dot-geoms.
line_size	Size of line-geoms.
panel	Logical, if TRUE, plots are arranged as panels; else, single plots for each diagnostic are returned.

Value

The data frame that is used for plotting.

Note

This function just prepares the data for plotting. To create the plots, **see** needs to be installed.

Examples

```
## Not run:
m <- lm(mpg ~ wt + cyl + gear + disp, data = mtcars)
check_model(m)

library(lme4)
m <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
check_model(m, panel = FALSE)

library(rstanarm)
m <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200)
check_model(m)

## End(Not run)
```

---

check_normality *Check model for (non-)normality of residuals.*

Description

Check model for (non-)normality of residuals.

Usage

```
check_normality(x, ...)
```

Arguments

x	A model object.
...	Currently not used.

Details

`check_normality()` calls `shapiro.test` and checks the standardized residuals for normal distribution. Note that this formal test almost always yields significant results for the distribution of residuals and visual inspection (e.g. Q-Q plots) are preferable.

Value

Invisibly returns the p-value of the test statistics. A p-value < 0.05 indicates a significant deviation from normal distribution

Examples

```
m <- lm(mpg ~ wt + cyl + gear + disp, data = mtcars)
check_normality(m)
```

check_outliers *Check for influential observations*

Description

Checks for and locates influential observations (i.e., "outliers") via Cook's Distance.

Usage

```
check_outliers(x, ...)

## Default S3 method:
check_outliers(x, threshold = 4/insight::n_obs(x), ...)
```

Arguments

x	A model object.
...	Currently not used.
threshold	The threshold indicating at which distance an observation is considered as outlier. For the Cook's Distance method, threshold defaults to 4 divided by numbers of observations.

Details

Performs a Cook's distance test to check for influential observations. Those greater than 4/n, are considered outliers. This relatively conservative threshold is useful only for detection, rather than justificaiton for automatic observation deletion. If users opt to drop observations that may be problematic, they may do so by specifying drop_outliers = TRUE.

Value

Check (message) on whether outliers were detected or not, as well as a data frame (with the original data that was used to fit the model), including information on the distance measure and whether or not an observation is considered as outlier.

References

Cook, R. D. (1977). Detection of influential observation in linear regression. Technometrics, 19(1), 15-18.

Examples

```
# select only mpg and disp (continuous)
mt1 <- mtcars[, c(1,3)]
# create some fake outliers and attach outliers to main df
mt2 <- rbind(mt1, data.frame(mpg = c(37, 40), disp = c(300, 400)))
# fit model with outliers
model <- lm(disp ~ mpg, data = mt2)
```

```
check_outliers(model)
```

`check_overdispersion` *Check overdispersion of GL(M)M's*

Description

`check_overdispersion()` checks generalized linear (mixed) models for overdispersion.

Usage

```
check_overdispersion(x, ...)
```

Arguments

`x`	Fitted model of class `merMod`, `glmmTMB`, `glm`, or `glm.nb` (package **MASS**).
`...`	Currently not used.

Details

A p-value < .05 indicates overdispersion. For `merMod`- and `glmmTMB`-objects, `check_overdispersion()` is based on the code in the GLMM FAQ, section *How can I deal with overdispersion in GLMMs?*. Note that this function only returns an *approximate* estimate of an overdispersion parameter, and is probably inaccurate for zero-inflated mixed models (fitted with `glmmTMB`). The same code is also used to check overdispersion for negative binomial models.

For Poisson-models, the overdispersion test is based on the code from *Gelman and Hill (2007), page 115.*

Overdispersion can be fixed by either modelling the dispersion parameter, or by choosing a different distributional family (like Quasi-Poisson, or negative binomial, see *Gelman and Hill (2007), pages 115-116*).

Value

A list with results from the overdispersion test, like chi-squared statistics, p-value or dispersion ratio.

References

- Bolker B et al. (2017): GLMM FAQ.
- Gelman, A., & Hill, J. (2007). Data analysis using regression and multilevel/hierarchical models. Cambridge; New York: Cambridge University Press.

Examples

```
library(glmmTMB)
data(Salamanders)

m <- glm(count ~ spp + mined, family = poisson, data = Salamanders)
check_overdispersion(m)

m <- glmmTMB(
  count ~ mined + spp + (1 | site),
  family = poisson,
  data = Salamanders
)
check_overdispersion(m)
```

---

check_singularity *Check mixed models for boundary fits*

Description

Check mixed models for boundary fits.

Usage

```
check_singularity(x, tolerance = 1e-05, ...)
```

Arguments

x	A mixed model.
tolerance	Indicates up to which value the convergence result is accepted. The larger tolerance is, the stricter the test will be.
...	Currently not used.

Details

If a model is "singular", this means that some dimensions of the variance-covariance matrix have been estimated as exactly zero. This often occurs for mixed models with complex random effects structures.

"While singular models are statistically well defined (it is theoretically sensible for the true maximum likelihood estimate to correspond to a singular fit), there are real concerns that (1) singular fits correspond to overfitted models that may have poor power; (2) chances of numerical problems and mis-convergence are higher for singular models (e.g. it may be computationally difficult to compute profile confidence intervals for such models); (3) standard inferential procedures such as Wald statistics and likelihood ratio tests may be inappropriate." (*lme4 Reference Manual*)

There is no gold-standard about how to deal with singularity and which random-effects specification to choose. Beside using fully Bayesian methods (with informative priors), proposals in a frequentist framework are:

- avoid fitting overly complex models, such that the variance-covariance matrices can be estimated precisely enough (*Matuschek et al. 2017*)
- use some form of model selection to choose a model that balances predictive accuracy and overfitting/type I error (*Bates et al. 2015*, *Matuschek et al. 2017*)
- "keep it maximal", i.e. fit the most complex model consistent with the experimental design, removing only terms required to allow a non-singular fit (*Barr et al. 2013*)

Value

`TRUE` if the model fit is singular.

References

- Bates D, Kliegl R, Vasishth S, Baayen H. Parsimonious Mixed Models. arXiv:1506.04967, June 2015.
- Barr DJ, Levy R, Scheepers C, Tily HJ. Random effects structure for confirmatory hypothesis testing: Keep it maximal. Journal of Memory and Language, 68(3):255-278, April 2013.
- Matuschek H, Kliegl R, Vasishth S, Baayen H, Bates D. Balancing type I error and power in linear mixed models. Journal of Memory and Language, 94:305-315, 2017.
- lme4 Reference Manual, https://cran.r-project.org/package=lme4

Examples

```
library(lme4)
data(sleepstudy)
set.seed(1)
sleepstudy$mygrp <- sample(1:5, size = 180, replace = TRUE)
sleepstudy$mysubgrp <- NA
for (i in 1:5) {
  filter_group <- sleepstudy$mygrp == i
  sleepstudy$mysubgrp[filter_group] <-
    sample(1:30, size = sum(filter_group), replace = TRUE)
}

model <- lmer(
  Reaction ~ Days + (1 | mygrp / mysubgrp) + (1 | Subject),
  data = sleepstudy
)

check_singularity(model)
```

check_zeroinflation *Check for zero-inflation in count models*

Description

check_zeroinflation() checks whether count models are over- or underfitting zeros in the outcome.

Usage

```
check_zeroinflation(x, tolerance = 0.05)
```

Arguments

x	Fitted model of class merMod, glmmTMB, glm, or glm.nb (package **MASS**).
tolerance	The tolerance for the ratio of observed and predicted zeros to considered as over- or underfitting zeros. A ratio between 1 +/- tolerance is considered as OK, while a ratio beyond or below this treshold would indicate over- or underfitting.

Details

If the amount of observed zeros is larger than the amount of predicted zeros, the model is underfitting zeros, which indicates a zero-inflation in the data. In such cases, it is recommended to use negative binomial or zero-inflated models.

Value

A list with information about the amount of predicted and observed zeros in the outcome, as well as the ratio between these two values.

Examples

```
library(glmmTMB)
data(Salamanders)
m <- glm(count ~ spp + mined, family = poisson, data = Salamanders)
check_zeroinflation(m)
```

classify_distribution *Machine learning model trained to classify distributions*

Description

Mean accuracy and Kappa of 0.86 and 0.85, repsectively.

Usage

```
classify_distribution
```

Format

An object of class `randomForest.formula` (inherits from `randomForest`) of length 8.

cronbachs_alpha *Cronbach's Alpha for Items or Scales*

Description

Compute various measures of internal consistencies for tests or item-scales of questionnaires.

Usage

```
cronbachs_alpha(x)
```

Arguments

x	A matrix or a data frame.

Details

The Cronbach's Alpha value for `x`. A value closer to 1 indicates greater internal consistency, where usually following rule of thumb is applied to interprete the results: $\alpha < 0.5$ is unacceptable, $0.5 < \alpha < 0.6$ is poor, $0.6 < \alpha < 0.7$ is questionable, $0.7 < \alpha < 0.8$ is acceptable, and everything > 0.8 is good or excellent.

Value

The Cronbach's Alpha value for `x`.

Examples

```
data(mtcars)
x <- mtcars[, c("cyl", "gear", "carb", "hp")]
cronbachs_alpha(x)
```

icc *Intraclass Correlation Coefficient (ICC)*

Description

This function calculates the intraclass-correlation coefficient (ICC) - sometimes also called *variance partition coefficient* (VPC) - for mixed effects models. The ICC is calculated for `merMod` (**lme4**), `glmmTMB` (**glmmTMB**), `MixMod` (**GLMMadpative**), `lme` (**nlme**), `mixed` (**afex**), and `stanreg` (**rstanarm**) objects. For models fitted with the **brms**-package, a variance decomposition based on the posterior predictive distribution is calculated (see 'Details').

Usage

```
icc(model, ...)

## S3 method for class 'brmsfit'
icc(model, re.form = NULL, robust = TRUE,
  ci = 0.95, ...)
```

Arguments

`model`	A mixed effects model of class `merMod`, `glmmTMB`, `MixMod`, `lme`, `mixed`, `stanreg` or `brmsfit`.
`...`	Currently not used.
`re.form`	Formula containing group-level effects to be considered in the prediction. If `NULL` (default), include all group-level effects. Else, for instance for nested models, name a specific group-level effect to calculate the variance decomposition for this group-level.
`robust`	Logical, if `TRUE`, the median instead of mean is used to calculate the central tendency of the variances.
`ci`	The Credible Interval level.

Details

Interpretation

The ICC can be interpreted as "the proportion of the variance explained by the grouping structure in the population". This index goes from 0, if the grouping conveys no information, to 1, if all observations in a group are identical (Gelman \& Hill, 2007, p. 258). In other word, the ICC "can also be interpreted as the expected correlation between two randomly drawn units that are in the same group" *(Hox 2010: 15)*, altough this definition might not apply to mixed models with more complex random effects structures.

Calculation

The ICC is calculated by dividing the random effect variance, σ_i^2, by the total variance, i.e. the

sum of the random effect variance and the residual variance, σ^2_ϵ.

Adjusted and conditional ICC

`icc()` calculates an adjusted and conditional ICC, which both take all sources of uncertainty (i.e. of *all random effects*) into account. While the *adjusted ICC* only relates to the random effects, the *conditional ICC* also takes the fixed effects variances into account (see *Nakagawa et al. 2017*). Typically, the *adjusted* ICC is of interest when the analysis of random effects is of interest. `icc()` returns a meaningful ICC also for more complex random effects structures, like models with random slopes or nested design (more than two levels) and is applicable for models with other distributions than Gaussian. For more details on the computation of the variances, see `get_variance`.

ICC for unconditional and conditional models

Usually, the ICC is calculated for the null model ("unconditional model"). However, according to *Raudenbush and Bryk (2002)* or *Rabe-Hesketh and Skrondal (2012)* it is also feasible to compute the ICC for full models with covariates ("conditional models") and compare how much, e.g., a level-2 variable explains the portion of variation in the grouping structure (random intercept).

ICC for specific group-levels

The proportion of variance for specific levels related to each other (e.g., similarity of level-1-units within level-2-units or level-2-units within level-3-units) must be calculated manually. Use `get_variance` to get the random intercept variances (between-group-variances) and residual variance of the model, and calculate the ICC for the various level correlations.

For example, for the ICC between level 1 and 2:

```
sum(insight::get_variance_intercept(model)) /
(sum(insight::get_variance_intercept(model)) + insight::get_variance_residual(model))
```

For for the ICC between level 2 and 3:

```
insight::get_variance_intercept(model)[2] /
sum(insight::get_variance_intercept(model))
```

ICC for brms-models

If `model` is of class `brmsfit`, `icc()` calculates a variance decomposition based on the posterior predictive distribution. In this case, first, the draws from the posterior predictive distribution *not conditioned* on group-level terms (`posterior_predict(..., re.form = NA)`) are calculated as well as draws from this distribution *conditioned* on *all random effects* (by default, unless specified else in `re.form`) are taken. Then, second, the variances for each of these draws are calculated. The "ICC" is then the ratio between these two variances. This is the recommended way to analyse random-effect-variances for non-Gaussian models. It is then possible to compare variances accross models, also by specifying different group-level terms via the `re.form`-argument.

Sometimes, when the variance of the posterior predictive distribution is very large, the variance ratio in the output makes no sense, e.g. because it is negative. In such cases, it might help to use `robust = TRUE`.

Value

A list with two values, the adjusted and conditional ICC. For models of class `brmsfit`, a list with two values, the decomposed ICC as well as the credible intervals for this ICC.

References

- Hox, J. J. (2010). Multilevel analysis: techniques and applications (2nd ed). New York: Routledge.
- Nakagawa, S., Johnson, P. C. D., & Schielzeth, H. (2017). The coefficient of determination R2 and intra-class correlation coefficient from generalized linear mixed-effects models revisited and expanded. Journal of The Royal Society Interface, 14(134), 20170213. doi: 10.1098/rsif.2017.0213
- Rabe-Hesketh, S., & Skrondal, A. (2012). Multilevel and longitudinal modeling using Stata (3rd ed). College Station, Tex: Stata Press Publication.
- Raudenbush, S. W., & Bryk, A. S. (2002). Hierarchical linear models: applications and data analysis methods (2nd ed). Thousand Oaks: Sage Publications.

Examples

```
library(lme4)
model <- lme4::lmer(Sepal.Length ~ Petal.Length + (1 | Species), data = iris)
icc(model)
```

---

item_difficulty *Difficulty of Questionnaire Items*

Description

Compute various measures of internal consistencies for tests or item-scales of questionnaires.

Usage

```
item_difficulty(x)
```

Arguments

`x`	Depending on the function, `x` may be a `matrix` as returned by the `cor`-function, or a data frame with items (e.g. from a test or questionnaire).

Details

This function calculates the item difficutly, which should range between 0.2 and 0.8. Lower values are a signal for more difficult items, while higher values close to one are a sign for easier items. The ideal value for item difficulty is `p + (1 - p) / 2`, where `p = 1 / max(x)`. In most cases, the ideal item difficulty lies between 0.5 and 0.8.

Value

A data frame with three columns: The name(s) of the item(s), the item difficulties for each item, and the ideal item difficulty.

Examples

```
data(mtcars)
x <- mtcars[, c("cyl", "gear", "carb", "hp")]
item_difficulty(x)
```

---

item_intercor *Mean Inter-Item-Correlation*

Description

Compute various measures of internal consistencies for tests or item-scales of questionnaires.

Usage

```
item_intercor(x, method = c("pearson", "spearman", "kendall"))
```

Arguments

x	A matrix as returned by the cor-function, or a data frame with items (e.g. from a test or questionnaire).
method	Correlation computation method. May be one of "spearman" (default), "pearson" or "kendall". You may use initial letter only.

Details

This function calculates a mean inter-item-correlation, i.e. a correlation matrix of x will be computed (unless x is already a matrix as returned by the cor()-function) and the mean of the sum of all item's correlation values is returned. Requires either a data frame or a computed cor()-object.

"Ideally, the average inter-item correlation for a set of items should be between .20 and .40, suggesting that while the items are reasonably homogenous, they do contain sufficiently unique variance so as to not be isomorphic with each other. When values are lower than .20, then the items may not be representative of the same content domain. If values are higher than .40, the items may be only capturing a small bandwidth of the construct." *(Piedmont 2014)*

Value

The mean inter-item-correlation value for x.

References

Piedmont RL. 2014. Inter-item Correlations. In: Michalos AC (eds) Encyclopedia of Quality of Life and Well-Being Research. Dordrecht: Springer, 3303-3304. doi: 10.1007/9789400707535_1493

Examples

```
data(mtcars)
x <- mtcars[, c("cyl", "gear", "carb", "hp")]
item_intercor(x)
```

---

item_reliability *Reliability Test for Items or Scales*

Description

Compute various measures of internal consistencies for tests or item-scales of questionnaires.

Usage

```
item_reliability(x, standardize = FALSE, digits = 3)
```

Arguments

x	A matrix or a data frame.
standardize	Logical, if TRUE, the data frame's vectors will be standardized. Recommended when the variables have different measures / scales.
digits	Amount of digits for returned values.

Details

This function calculates the item discriminations (corrected item-total correlations for each item of x with the remaining items) and the Cronbach's alpha for each item, if it was deleted from the scale. The absolute value of the item discrimination indices should be above 0.1. An index between 0.1 and 0.3 is considered as "fair", while an index above 0.3 (or below -0.3) is "good". Items with low discrimination indices are often ambiguously worded and should be examined. Items with negative indices should be examined to determine why a negative value was obtained (e.g. reversed answer categories regarding positive and negative poles).

Value

A data frame with the corrected item-total correlations (*item discrimination*, column item_discrimination) and Cronbach's Alpha (if item deleted, column alpha_if_deleted) for each item of the scale, or NULL if data frame had too less columns.

Examples

```
data(mtcars)
x <- mtcars[, c("cyl", "gear", "carb", "hp")]
item_reliability(x)
```

`item_split_half` *Split-Half Reliability*

Description

Compute various measures of internal consistencies for tests or item-scales of questionnaires.

Usage

```
item_split_half(x, digits = 3)
```

Arguments

`x`	A matrix or a data frame.
`digits`	Amount of digits for returned values.

Details

This function calculates the split-half reliability for items in `x`, including the Spearman-Brown adjustment. Splitting is done by selecting odd versus even columns in `x`. A value closer to 1 indicates greater internal consistency.

Value

A list with two elements: the split-half reliability `splithalf` and the Spearman-Brown corrected split-half reliability `spearmanbrown`.

References

Spearman C. 1910. Correlation calculated from faulty data. British Journal of Psychology (3): 271-295. doi: 10.1111/j.20448295.1910.tb00206.x

Brown W. 1910. Some experimental results in the correlation of mental abilities. British Journal of Psychology (3): 296-322. doi: 10.1111/j.20448295.1910.tb00207.x

Examples

```
data(mtcars)
x <- mtcars[, c("cyl", "gear", "carb", "hp")]
item_split_half(x)
```

looic *LOO-related Indices for Bayesian regressions.*

Description

Compute LOOIC (leave-one-out cross-validation (LOO) information criterion) and ELPD (expected log predictive density) for Bayesian regressions.

Usage

```
looic(model)
```

Arguments

model	A Bayesian regression model.

Value

A list with four elements, the ELPD, LOOIC and their standard errors.

Examples

```
library(rstanarm)

model <- stan_glm(mpg ~ wt + cyl, data = mtcars, chains = 1, iter = 500)
looic(model)
```

model_performance *Model Performance*

Description

See the documentation for your object's class: mixed models, Bayesian models and all other models. `compare_performance()` computes indices of model performance for different models at once and hence allows comparison of indices accros models.

Usage

```
model_performance(model, ...)

compare_performance(..., metrics = "all")
```

Arguments

model	Statistical model.
...	Arguments passed to or from other methods, resp. for compare_performance(), one or multiple model objects (also of different classes).
metrics	Can be "all" or a character vector of metrics to be computed. See related documentation of object's class for details.

Details

If all models are of the same class, compare_performance() returns an additional column named BF, which shows the Bayes factor (see bayesfactor_models) for each model against the denominator model. The *first* model is used as denominator model, and its Bayes factor is set to NA to indicate the reference model.

Value

For model_performance(), a data frame (with one row) and one column per "index" (see metrics). For compare_performance(), the same data frame with one row per model.

Examples

```
library(lme4)

m1 <- lm(mpg ~ wt + cyl, data = mtcars)
model_performance(m1)

m2 <- glm(vs ~ wt + mpg, data = mtcars, family = "binomial")
m3 <- lmer(Petal.Length ~ Sepal.Length + (1 | Species), data = iris)
compare_performance(m1, m2, m3)

data(iris)
lm1 <- lm(Sepal.Length ~ Species, data = iris)
lm2 <- lm(Sepal.Length ~ Species + Petal.Length, data = iris)
lm3 <- lm(Sepal.Length ~ Species * Petal.Length, data = iris)
compare_performance(lm1, lm2, lm3)
```

---

model_performance.lm *Performance of Regression Models*

Description

Compute indices of model performance for regression models.

Usage

```
## S3 method for class 'lm'
model_performance(model, metrics = "all", verbose = TRUE,
  ...)

## S3 method for class 'glm'
model_performance(model, metrics = "all", verbose = TRUE,
  ...)
```

Arguments

model	A model.
metrics	Can be "all" or a character vector of metrics to be computed (some of c("AIC", "BIC", "R2", "RMSE",
verbose	Toggle off warnings.
...	Arguments passed to or from other methods.

Details

Depending on model, following indices are computed:

- **AIC** Akaike's Information Criterion, see AIC
- **BIC** Bayesian Information Criterion, see BIC
- **R2** r-squared value, see r2
- **R2_adj** adjusted r-squared, see r2
- **RMSE** root mean squared error, see performance_rmse
- **LOGLOSS** Log-loss, see performance_logloss
- **SCORE_LOG** score of logarithmic proper scoring rule, see performance_score
- **SCORE_SPHERICAL** score of spherical proper scoring rule, see performance_score
- **PCP** percentage of correct predictions, see performance_pcp

Value

A data frame (with one row) and one column per "index" (see metrics).

Examples

```
model <- lm(mpg ~ wt + cyl, data = mtcars)
model_performance(model)

model <- glm(vs ~ wt + mpg, data = mtcars, family = "binomial")
model_performance(model)
```

model_performance.merMod
Performance of Mixed Models

Description

Compute indices of model performance for mixed models.

Usage

```
## S3 method for class 'merMod'
model_performance(model, metrics = "all", ...)
```

Arguments

model	Object of class merMod, glmmTMB, lme or MixMod.
metrics	Can be "all" or a character vector of metrics to be computed (some of c("AIC", "BIC", "R2", "ICC",
...	Arguments passed to or from other methods.

Details

This method returns the *adjusted ICC* only, as this is typically of interest when judging the variance attributed to the random effects part of the model (see also icc).

Furthermore, see 'Details' in model_performance.lm for more details on returned indices.

Value

A data frame (with one row) and one column per "index" (see metrics).

Examples

```
library(lme4)
model <- lmer(Petal.Length ~ Sepal.Length + (1 | Species), data = iris)
model_performance(model)
```

model_performance.stanreg
Performance of Bayesian Models

Description

Compute indices of model performance for (general) linear models.

Usage

```
## S3 method for class 'stanreg'
model_performance(model, metrics = "all", ...)
```

Arguments

model	Object of class stanreg or brmsfit.
metrics	Can be "all" or a character vector of metrics to be computed (some of c("LOOIC", "WAIC", "R2", "R2_
...	Arguments passed to or from other methods.

Details

See 'Details' in model_performance.lm for more details on returned indices.

Value

A data frame (with one row) and one column per "index" (see metrics).

References

Gelman, A., Goodrich, B., Gabry, J., & Vehtari, A. (2018). R-squared for Bayesian regression models. The American Statistician, The American Statistician, 1-6.

See Also

r2_bayes

Examples

```
library(rstanarm)

model <- stan_glm(mpg ~ wt + cyl, data = mtcars, chains = 1, iter = 500)
model_performance(model)

model <- stan_glmer(
  mpg ~ wt + cyl + (1 | gear),
  data = mtcars,
  chains = 1,
  iter = 500
```

```
)
model_performance(model)
```

`performance_accuracy` *Accuracy of predictions from model fit*

Description

This function calculates the predictive accuracy of linear or logistic regression models.

Usage

```
performance_accuracy(model, method = c("cv", "boot"), k = 5,
  n = 1000)
```

Arguments

`model`	Fitted model object of class `lm` or `glm`, the latter being a logistic regression model (binary response).
`method`	Character string, indicating whether crossvalidation (`method = "cv"`) or bootstrapping (`method = "boot"`) is used to compute the accuracy values.
`k`	The number of folds for the kfold-crossvalidation.
`n`	Number of bootstrap-samples.

Details

For linar models, the accuracy is the correlation coefficient between the actual and the predicted value of the outcome. For logistic regression models, the accuracy corresponds to the AUC-value, calculated with the auc-function.

The accuracy is the mean value of multiple correlation resp. AUC-values, which are either computed with crossvalidation or nonparametric bootstrapping (see argument `method`). The standard error is the standard deviation of the computed correlation resp. AUC-values.

Value

A list with two values: The `accuracy` of the model predictions, i.e. the proportion of accurately predicted values from the model and its standard error, `std.error`.

Examples

```
model <- lm(mpg ~ wt + cyl, data = mtcars)
performance_accuracy(model)

model <- glm(vs ~ wt + mpg, data = mtcars, family = "binomial")
performance_accuracy(model)
```

performance_aicc *Compute second order AIC*

Description

Compute the second-order Akaike's information criterion (AICc). The second-order (or small sample) is a AIC with a correction for small sample sizes.

Usage

```
performance_aicc(x, ...)
```

Arguments

x	A model object.
...	Currently not used.

Value

Numeric, the AICc value.

References

- Akaike, H. (1973) Information theory as an extension of the maximum likelihood principle. In: Second International Symposium on Information Theory, pp. 267–281. Petrov, B.N., Csaki, F., Eds, Akademiai Kiado, Budapest.
- Hurvich, C. M., Tsai, C.-L. (1991) Bias of the corrected AIC criterion for underfitted regression and time series models. Biometrika 78, 499–509.

Examples

```
m <- lm(mpg ~ wt + cyl + gear + disp, data = mtcars)
AIC(m)
performance_aicc(m)
```

performance_hosmer *Hosmer-Lemeshow goodness-of-fit test*

Description

Check model quality of logistic regression models.

Usage

```
performance_hosmer(model, n_bins = 10)
```

Arguments

model	A glm-object with binomial-family.
n_bins	Numeric, the number of bins to divide the data.

Details

A well-fitting model shows *no* significant difference between the model and the observed data, i.e. the reported p-value should be greater than 0.05.

Value

An object of class hoslem_test with following values: chisq, the Hosmer-Lemeshow chi-squared statistic; df, degrees of freedom and p.value the p-value for the goodness-of-fit test.

References

Hosmer, D. W., & Lemeshow, S. (2000). Applied Logistic Regression. Hoboken, NJ, USA: John Wiley & Sons, Inc. doi: 10.1002/0471722146

Examples

```
model <- glm(vs ~ wt + mpg, data = mtcars, family = "binomial")
performance_hosmer(model)
```

---

performance_logloss *Log Loss*

Description

Compute the log loss for models with binary outcome.

Usage

```
performance_logloss(model, ...)
```

Arguments

model	Model with binary outcome.
...	Currently not used.

Details

Logistic regression models predict the probability of an outcome of being a "success" or "failure" (or 1 and 0 etc.). performance_logloss() evaluates how good or bad the predicted probabilities are. High values indicate bad predictions, while low values indicate good predictions. The lower the log-loss, the better the model predicts the outcome.

Value

Numeric, the log loss of `model`.

See Also

`performance_score()`

Examples

```
data(mtcars)
m <- glm(formula = vs ~ hp + wt, family = binomial, data = mtcars)
performance_logloss(m)
```

`performance_mse` *Mean Square Error of Linear Models*

Description

Compute mean square error of linear models.

Usage

```
performance_mse(model, verbose = TRUE)

mse(model, verbose = TRUE)
```

Arguments

`model`	A model.
`verbose`	Toggle off warnings.

Details

The mean square error is the mean of the sum of squared residuals, i.e. it measures the average of the squares of the errors. Lower values (closer to zero) indicate better fit.

Value

Numeric, the mean square error of `model`.

Examples

```
data(mtcars)
m <- lm(mpg ~ hp + gear, data = mtcars)
performance_mse(m)
```

`performance_pcp` *Percentage of Correct Predictions*

Description

Percentage of correct predictions (PCP) for models with binary outcome.

Usage

```
performance_pcp(model, ci = 0.95, method = "Herron")
```

Arguments

`model`	Model with binary outcome.
`ci`	The level of the confidence interval.
`method`	Name of the method to calculate the PCP (see 'Details'). Default is `"Herron"`. May be abbreviated.

Details

`method = "Gelman-Hill"` (or `"gelman_hill"`) computes the PCP based on the proposal from *Gelman and Hill 2017, 99*, which is defined as the proportion of cases for which the deterministic prediction is wrong, i.e. the proportion where the predicted probability is above 0.5, although y=0 (and vice versa) (see also *Herron 1999, 90*).

`method = "Herron"` (or `"herron"`) computes a modified version of the PCP (*Herron 1999, 90-92*), which is the sum of predicted probabilities, where y=1, plus the sum of 1 - predicted probabilities, where y=0, divided by the number of observations. This approach is said to be more accurate.

The PCP ranges from 0 to 1, where values closer to 1 mean that the model predicts the outcome better than models with an PCP closer to 0. In general, the PCP should be above 0.5 (i.e. 50%), the closer to one, the better. Furthermore, the PCP of the full model should be considerably above the null model's PCP.

The likelihood-ratio test indicates whether the model has a significantly better fit than the null-model (in such cases, $p < 0.05$).

Value

A list with several elements: the percentage of correct predictions of the full and the null model, their confidence intervals, as well as the chi-squared and p-value from the Likelihood-Ratio-Test between the full and null model.

References

- Herron, M. (1999). Postestimation Uncertainty in Limited Dependent Variable Models. Political Analysis, 8, 83–98.
- Gelman, A., & Hill, J. (2007). Data analysis using regression and multilevel/hierarchical models. Cambridge; New York: Cambridge University Press, 99

Examples

```
data(mtcars)
m <- glm(formula = vs ~ hp + wt, family = binomial, data = mtcars)
performance_pcp(m)
performance_pcp(m, method = "Gelman-Hill")
```

---

performance_rmse *Root Mean Squared Error*

Description

Compute root mean squared error for (mixed effects) models, including Bayesian regression models.

Usage

```
performance_rmse(model, normalized = FALSE, verbose = TRUE)

rmse(model, normalized = FALSE, verbose = TRUE)
```

Arguments

model	A model.
normalized	Logical, use TRUE if normalized rmse should be returned.
verbose	Toggle off warnings.

Details

The RMSE is the square root of the variance of the residuals and indicates the absolute fit of the model to the data (difference between observed data to model's predicted values). It can be interpreted as the standard deviation of the unexplained variance, and is in the same units as the response variable. Lower values indicate better model fit.

The normalized RMSE is the proportion of the RMSE related to the range of the response variable. Hence, lower values indicate less residual variance.

Value

Numeric, the root mean squared error.

Examples

```
library(nlme)
m <- lme(distance ~ age, data = Orthodont)

# RMSE
performance_rmse(m, normalized = TRUE)

# normalized RMSE
performance_rmse(m, normalized = TRUE)
```

`performance_roc` *Simple ROC curve*

Description

This function calculates a simple ROC curves of x/y coordinates based on response and predictions of a binomial model.

Usage

```
performance_roc(x, ..., predictions, new_data)
```

Arguments

`x`	A numeric vector, representing the outcome (0/1), or a model with binomial outcome.
`...`	One or more models with binomial outcome. In this case, `new_data` is ignored.
`predictions`	If `x` is numeric, a numeric vector of same length as `x`, representing the actual predicted values.
`new_data`	If `x` is a model, a data frame that is passed to `predict()` as `newdata`-argument. If `NULL`, the ROC for the full model is calculated.

Value

A data frame with three columns, the x/y-coordinate pairs for the ROC curve (`Sensivity` and `Specifity`), and a column with the model name.

Examples

```
library(bayestestR)
data(iris)

set.seed(123)
iris$y <- rbinom(nrow(iris), size = 1, .3)
folds <- sample(nrow(iris), size = nrow(iris) / 8, replace = FALSE)
test_data <- iris[folds, ]
```

```
train_data <- iris[-folds, ]

model <- glm(y ~ Sepal.Length + Sepal.Width, data = train_data, family = "binomial")
performance_roc(model, new_data = test_data)

roc <- performance_roc(model, new_data = test_data)
area_under_curve(roc$Sensivity, roc$Specifity)

m1 <- glm(y ~ Sepal.Length + Sepal.Width, data = iris, family = "binomial")
m2 <- glm(y ~ Sepal.Length + Petal.Width, data = iris, family = "binomial")
m3 <- glm(y ~ Sepal.Length + Species, data = iris, family = "binomial")
performance_roc(m1, m2, m3)
```

`performance_rse` *Residual Standard Error for Linear Models*

Description

Compute residual standard error of linear models.

Usage

```
performance_rse(model)
```

Arguments

`model`	A model.

Details

The residual standard error is the square root of the residual sum of squares divided by the residual degrees of freedom.

Value

Numeric, the residual standard error of `model`.

Examples

```
data(mtcars)
m <- lm(mpg ~ hp + gear, data = mtcars)
performance_rse(m)
```

`performance_score` *Proper Scoring Rules*

Description

Calculates the logarithmic, quadratic/Brier and spherical score from a model with binary or count outcome.

Usage

```
performance_score(model, verbose = TRUE)
```

Arguments

`model`	Model with binary or count outcome.
`verbose`	Toggle off warnings.
`...`	Currently not used.

Details

Proper scoring rules can be used to evaluate the quality of model predictions and model fit. `performance_score()` calculates the logarithmic, quadratic/Brier and spherical scoring rules. The spherical rule takes values in the interval `[0, 1]`, with values closer to 1 indicating a more accurate model, and the logarithmic rule in the interval `[-Inf, 0]`, with values closer to 0 indicating a more accurate model.

Value

A list with three elements, the logarithmic, quadratic/Brier and spherical score.

Note

Code is partially based on GLMMadaptive::scoring_rules().

References

Carvalho, A. (2016). An overview of applications of proper scoring rules. Decision Analysis 13, 223–242. doi: 10.1287/deca.2016.0337

See Also

`performance_logloss()`

Examples

```
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18, 17, 15, 20, 10, 20, 25, 13, 12)
outcome <- gl(3, 1, 9)
treatment <- gl(3, 3)
model <- glm(counts ~ outcome + treatment, family = poisson())

performance_score(model)

## Not run:
library(glmmTMB)
data(Salamanders)
model <- glmmTMB(
  count ~ spp + mined + (1 | site),
  zi =  ~ spp + mined,
  family = nbinom2(),
  data = Salamanders
)

performance_score(model)

## End(Not run)
```

---

principal_components *Principal Components Analysis*

Description

This function performs a principal component analysis and returns the loadings (of the unrotated matrix) as data frame, or returns a rotated matrix of the loadings (if `rotation` is not `NULL`).

Usage

```
principal_components(x, rotation = NULL, n_comp = NULL)
```

Arguments

`x`	A data frame or a `prcomp`-object.
`rotation`	Rotation of the factor loadings. May be one of `"varimax"`, `"quartimax"`, `"promax"`, `"oblimin"`, `"si` or `"none"`. If `rotatoin = NULL`, loadings for the principal components from the unrotated matrix are returned.
`n_comp`	Number of components to extract. If `rotation = "varmiax"` and `n_comp = NULL`, number of components is based on the Kaiser-criteria.

Details

The `print()`-method has a `cutoff`-argument, which is a scalar between 0 and 1, indicating which (absolute) values from the *rotated* loadings (i.e. when `rotation` is *not* `NULL`) should be blank in the output. By default, all loadings between -.1 and .1 are not shown.

Value

If `rotation = NULL`, a data frame with all loadings of principal components. Else, a rotated loadings matrix, as data frame. Details on the variance components are saved as attributes.

Examples

```
data(iris)
principal_components(iris[, 1:4])

data(iris)
principal_components(iris[, 1:4], rotation = "varimax", n_comp = 2)

pr <- principal_components(iris[, 1:4], rotation = "varimax", n_comp = 2)

# show all
print(pr, cutoff = .001)

# show only some
print(pr, cutoff = .5)
```

---

r2 *Compute the model's R2*

Description

Calculate the R2 value for different model objects. Depending on the model, R2, pseudo-R2 or marginal / adjusted R2 values are returned.

Usage

```
r2(model, ...)
```

Arguments

`model`	A statistical model.
`...`	Currently not used.

Value

Returns a list containing values related to the most appropriate R2 for the given model. See the list below:

- Logistic models: Tjur's R2
- General linear models: Nagelkerke's R2
- Multinomial Logit: McFadden's R2
- Models with zero-inflation: R2 for zero-inflated models
- Mixed models: Nakagawa's R2
- Bayesian models: R2 bayes

See Also

`r2_bayes`, `r2_coxsnell`, `r2_kl`, `r2_loo`, `r2_mcfadden`, `r2_nagelkerke`, `r2_nakagawa`, `r2_tjur`, `r2_xu` and `r2_zeroinflated`.

Examples

```
model <- glm(vs ~ wt + mpg, data = mtcars, family = "binomial")
r2(model)

library(lme4)
model <- lmer(Sepal.Length ~ Petal.Length + (1 | Species), data = iris)
r2(model)
```

---

r2_bayes *Bayesian R2*

Description

Compute R2 for Bayesian models. For mixed models (including a random part), it additionally computes the R2 related to the fixed effects only (marginal R2).

Usage

```
r2_bayes(model, robust = TRUE)
```

Arguments

`model`	A Bayesian regression model.
`robust`	Logical, if `TRUE`, the median instead of mean is used to calculate the central tendency of the variances.

Details

`r2_bayes()` returns an "unadjusted" R2 value. See `r2_loo` to calculate a LOO-adjusted R2, which comes conceptionally closer to an adjusted R2 measure.

For mixed models, the conditional and marginal R2 are returned. The marginal R2 considers only the variance of the fixed effects, while the conditional R2 takes both the fixed and random effects into account.

Value

A list with the Bayesian R2 value. For mixed models, a list with the Bayesian R2 value and the marginal Bayesian R2 value. The standard errors for the R2 values are saved as attributes.

References

Gelman, A., Goodrich, B., Gabry, J., & Vehtari, A. (2018). R-squared for Bayesian regression models. The American Statistician, 1–6. doi: 10.1080/00031305.2018.1549100

Examples

```
library(rstanarm)

model <- stan_glm(mpg ~ wt + cyl, data = mtcars, chains = 1, iter = 500)
r2_bayes(model)

model <- stan_lmer(
  Petal.Length ~ Petal.Width + (1 | Species),
  data = iris,
  chains = 1,
  iter = 500
)
r2_bayes(model)

## Not run:
library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
r2_bayes(model)

model <- brms::brm(Petal.Length ~ Petal.Width + (1 | Species), data = iris)
r2_bayes(model)

## End(Not run)
```

`r2_coxsnell` *Cox & Snell's R2*

Description

Calculates the pseudo-R2 value based on the proposal from *Cox & Snell (1989)*.

Usage

```
r2_coxsnell(model)
```

Arguments

`model`	Model with binary outcome.

Details

This index was proposed by *Cox & Snell (1989, pp. 208-9)* and, apparently independently, by *Magee (1990)*; but had been suggested earlier for binary response models by *Maddala (1983)*. However, this index achieves a maximum of less than 1 for discrete models (i.e. models whose likelihood is a product of probabilities) which have a maximum of 1, instead of densities, which can become infinite *(Nagelkerke, 1991)*.

Value

A named vector with the R2 value.

References

- Cox, D. R., Snell, E. J. (1989). Analysis of binary data (Vol. 32). Monographs on Statistics and Applied Probability.
- Magee, L. (1990). R 2 measures based on Wald and likelihood ratio joint significance tests. The American Statistician, 44(3), 250-253.
- Maddala, G. S. (1986). Limited-dependent and qualitative variables in econometrics (No. 3). Cambridge university press.
- Nagelkerke, N. J. (1991). A note on a general definition of the coefficient of determination. Biometrika, 78(3), 691-692.

Examples

```
model <- glm(vs ~ wt + mpg, data = mtcars, family = "binomial")
r2_coxsnell(model)
```

r2_kl *Kullback-Leibler R2*

Description

Calculates the Kullback-Leibler-divergence-based R2 for generalized linear models.

Usage

```
r2_kl(model, adjust = TRUE)
```

Arguments

model	A generalized linear model.
adjust	Logical, if TRUE (the default), the adjusted R2 value is returned.

Value

A named vector with the R2 value.

References

Cameron, A. C. and Windmeijer, A. G. (1997) An R-squared measure of goodness of fit for some common nonlinear regression models. Journal of Econometrics, 77: 329-342.

Examples

```
model <- glm(vs ~ wt + mpg, data = mtcars, family = "binomial")
r2_kl(model)
```

r2_loo *LOO-adjusted R2*

Description

Compute LOO-adjusted R2.

Usage

```
r2_loo(model)
```

Arguments

model	A Bayesian regression model.

Details

Unlike r2_bayes, which returns an "unadjusted" R2 value, r2_loo() calculates a LOO-adjusted R2, which comes conceptionally closer to an "adjusted" R2 measure.

Value

The LOO-adjusted R2 for model, as numeric value.

Examples

```
library(rstanarm)

model <- stan_glm(mpg ~ wt + cyl, data = mtcars, chains = 1, iter = 500)
r2_loo(model)
```

---

r2_mcfadden *McFadden's R2*

Description

Calculates McFadden's pseudo R2.

Usage

```
r2_mcfadden(model)
```

Arguments

model	Generalized linear or multinomial logit (mlogit) model.

Value

For most models, a list with McFadden's R2 and adjusted McFadden's R2 value. For some models, only McFadden's R2 is available.

References

- McFadden, D. (1987). Regression-based specification tests for the multinomial logit model. Journal of econometrics, 34(1-2), 63-82.
- McFadden, D. (1973). Conditional logit analysis of qualitative choice behavior.

Examples

```
library(mlogit)
data("Fishing", package = "mlogit")
Fish <- mlogit.data(Fishing, varying = c(2:9), shape = "wide", choice = "mode")

model <- mlogit(mode ~ price + catch, data = Fish)
r2_mcfadden(model)
```

`r2_mckelvey` *McKelvey & Zavoinas R2*

Description

Calculates McKelvey & Zavoinas pseudo R2.

Usage

```
r2_mckelvey(model)
```

Arguments

`model`	Generalized linear model.

Details

McKelvey & Zavoinas R2 is based on the explained variance, where the variance of the predicted response is divided by the sum of the variance of the predicted response and residual variance. For binomial models, the residual variance is either `pi^2/3` for logit-link and 1 for probit-link. For poisson-models, the residual variance is based on log-normal approximation, similar to the *distribution-specific variance* as described in `get_variance`.

Value

The R2 value.

References

- McKelvey, R., Zavoina, W. (1975), "A Statistical Model for the Analysis of Ordinal Level Dependent Variables", Journal of Mathematical Sociology 4, S. 103–120.

Examples

```
## Dobson (1990) Page 93: Randomized Controlled Trial:
counts <- c(18,17,15,20,10,20,25,13,12)#
outcome <- gl(3,1,9)
treatment <- gl(3,3)
model <- glm(counts ~ outcome + treatment, family = poisson())

r2_mckelvey(model)
```

`r2_nagelkerke` *Nagelkerke's R2*

Description

Calculate Nagelkerke's pseudo-R2.

Usage

```
r2_nagelkerke(model)
```

Arguments

`model`	A generalized linear model, including cumulative links resp. multinomial models.

Value

A named vector with the R2 value.

References

Nagelkerke, N. J. (1991). A note on a general definition of the coefficient of determination. Biometrika, 78(3), 691-692.

Examples

```
model <- glm(vs ~ wt + mpg, data = mtcars, family = "binomial")
r2_nagelkerke(model)
```

r2_nakagawa *Nakagawa's R2 for mixed models*

Description

Compute the marginal and conditional r-squared value for mixed effects models with complex random effects structures.

Usage

```
r2_nakagawa(model)
```

Arguments

model	A mixed effects model.

Details

Marginal and conditional r-squared values for mixed models are calculated based on *Nakagawa et al. 2017*. For more details on the computation of the variances, see `get_variance`.

The marginal r-squared considers only the variance of the fixed effects, while the conditional r-squared takes both the fixed and random effects into account. The random effect variances are actually the mean random effect variances, thus the r-squared value is also appropriate for mixed models with random slopes or nested random effects (see *Johnson 2014*).

Value

A list with the conditional and marginal R2 values.

References

- Johnson, P. C. D. (2014). Extension of Nakagawa & Schielzeth's R2 GLMM to random slopes models. Methods in Ecology and Evolution, 5(9), 944–946. doi: 10.1111/2041210X.12225
- Nakagawa, S., & Schielzeth, H. (2013). A general and simple method for obtaining R2 from generalized linear mixed-effects models. Methods in Ecology and Evolution, 4(2), 133–142. doi: 10.1111/j.2041210x.2012.00261.x
- Nakagawa, S., Johnson, P. C. D., & Schielzeth, H. (2017). The coefficient of determination R2 and intra-class correlation coefficient from generalized linear mixed-effects models revisited and expanded. Journal of The Royal Society Interface, 14(134), 20170213. doi: 10.1098/rsif.2017.0213

Examples

```
library(lme4)
model <- lmer(Sepal.Length ~ Petal.Length + (1 | Species), data = iris)
r2_nakagawa(model)
```

r2_tjur *Tjur's R2 - coefficient of determination (D)*

Description

This method calculates the Coefficient of Discrimination `D` (also known as Tjur's R2; *Tjur, 2009*) for generalized linear (mixed) models for binary outcomes. It is an alternative to other pseudo-R2 values like Nagelkerke's R2 or Cox-Snell R2. The Coefficient of Discrimination `D` can be read like any other (pseudo-)R2 value.

Usage

```
r2_tjur(model)
```

Arguments

`model`	Binomial Model.

Value

A named vector with the R2 value.

References

Tjur, T. (2009). Coefficients of determination in logistic regression models - A new proposal: The coefficient of discrimination. The American Statistician, 63(4), 366-372.

Examples

```
model <- glm(vs ~ wt + mpg, data = mtcars, family = "binomial")
r2_tjur(model)
```

r2_xu *Xu' R2 (Omega-squared)*

Description

Calculates Xu' Omega-squared value, a simple R2 equivalent for linear mixed models.

Usage

```
r2_xu(model)
```

Arguments

`model`	A linear (mixed) model.

Details

`r2_xu()` is a crude measure for the explained variance from linear (mixed) effects models, which is originally denoted as Ω^2.

Value

The R2 value.

References

Xu, R. (2003). Measuring explained variation in linear mixed effects models. Statistics in Medicine, 22(22), 3527–3541. doi: 10.1002/sim.1572

Examples

```
model <- lm(Sepal.Length ~ Petal.Length + Species, data = iris)
r2_xu(model)
```

`r2_zeroinflated` *R2 for models with zero-inflation*

Description

Calculates R2 for models with zero-inflation component, including mixed effects models.

Usage

```
r2_zeroinflated(model, method = c("default", "correlation"))
```

Arguments

`model`	A model.
`method`	Indicates the method to calculate R2. See 'Details'. May be abbreviated.

Details

The default-method calculates an R2 value based on the residual variance divided by the total variance. For `method = "correlation"`, R2 is a correlation-based measure, which is rather crude. It simply computes the squared correlation between the model's actual and predicted reponse.

Value

For the default-method, a list with the R2 and adjusted R2 values. For `method = "correlation"`, a named numeric vector with the correlation-based R2 value.

Examples

```
library(pscl)
data(bioChemists)
model <- zeroinfl(
  art ~ fem + mar + kid5 + ment | kid5 + phd,
  data = bioChemists
)

r2_zeroinflated(model)
```

Index