

Package ‘sjstats’

September 26, 2016

Type Package

Encoding UTF-8

Title Collection of Convenient Functions for Common Statistical Computations

Version 0.5.0

Date 2016-09-26

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

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

Description Collection of convenient functions for common statistical computations, which are not directly provided by R's base or stats packages. This package aims at providing, first, shortcuts for statistical measures, which otherwise could only be calculated with additional effort (like standard errors or root mean squared errors). Second, these shortcut functions are generic (if appropriate), and can be applied not only to vectors, but also to other objects as well (e.g., the Coefficient of Variation can be computed for vectors, linear models, or linear mixed models; the `r2()`-function returns the r-squared value for 'lm', 'glm', 'merMod' or 'lme' objects). The focus of most functions lies on summary statistics or fit measures for regression models, including generalized linear models and mixed effects models. However, some of the functions also deal with other statistical measures, like Cronbach's Alpha, Cramer's V, Phi etc.

License GPL-3

Depends R (>= 3.2), stats, utils

Imports coin, dplyr (>= 0.5.0), lme4 (>= 1.1-10), MASS, Matrix, nlme, sjmisc (>= 2.0.0), tidyr (>= 0.6.0), tibble (>= 1.2.0)

Suggests AER, arm, ggplot2, graphics, Hmisc, lmerTest, pbkrtest (>= 0.4-6), pwr

URL <https://github.com/sjPlot/sjstats>

BugReports <https://github.com/sjPlot/sjstats/issues>

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-09-26 14:24:01

R topics documented:

sjstats-package	3
bootstrap	4
boot_ci	5
chisq_gof	7
cod	8
converge_ok	9
cramer	10
cronb	11
cv	12
deff	13
efc	14
eta_sq	15
get_model_pval	16
get_re_var	17
hoslem_gof	18
icc	19
inequ_trend	22
levene_test	23
mean_n	24
merMod_p	25
mic	26
mwu	27
odds_to_rr	28
overdisp	29
phi	30
pred_vars	31
r2	32
reliab_test	34
re_var	36
rmse	37
rse	38
se	39
se_ybar	40
smpsize_lmm	41
std	42
std_beta	43
table_values	45
weight	46
weight2	47
wtd_sd	48

sjstats-package *Collection of Convenient Functions for Common Statistical Computations*

Description

Collection of convenient functions for common statistical computations, which are not directly provided by R’s base or stats packages. This package aims at providing, first, shortcuts for statistical measures, which otherwise could only be calculated with additional effort (like standard errors or root mean squared errors). Second, these shortcut functions are generic (if appropriate), and can be applied not only to vectors, but also to other objects as well (e.g., the Coefficient of Variation can be computed for vectors, linear models, or linear mixed models; the r2-function returns the r-squared value for lm, glm, merMod or lme objects). The focus of most functions lies on summary statistics or fit measures for regression models, including generalized linear models and mixed effects models. However, some of the functions deal with other statistical measures, like Cronbach’s Alpha, Cramer’s V, Phi etc.

The comprised tools include:

- For regression and mixed models: Coefficient of Variation, Root Mean Squared Error, Residual Standard Error, Coefficient of Discrimination, R-squared and pseudo-R-squared values, standardized beta values
- Especially for mixed models: Design effect, ICC, sample size calculation, convergence and overdispersion tests

Other statistics:

- Cramer’s V, Cronbach’s Alpha, Mean Inter-Item-Correlation, Mann-Whitney-U-Test, Item-scale reliability tests

Details

Package: sjstats
 Type: Package
 Version: 0.5.0
 Date: 2016-09-26
 License: GPL-3

Author(s)

Daniel Lüdtke <d.luedecke@uke.de>

References

- Documentation: <http://www.strengjacke.de/sjPlot>
- Github: <https://github.com/sjPlot/sjstats>
- Bug reports: <https://github.com/sjPlot/sjstats/issues>

bootstrap

Generate nonparametric bootstrap replications

Description

Generates n bootstrap samples of data and returns the bootstrapped data frames as list-variable.

Usage

```
bootstrap(data, n, size)
```

Arguments

data	A data frame.
n	Number of bootstraps to be generated
size	Optional, size of the bootstrap samples. May either be a number between 1 and <code>nrow(data)</code> or a value between 0 and 1 to sample a proportion of observations from data (see 'Examples').

Details

By default, each bootstrap sample has the same number of observations as data. To generate bootstrap samples without resampling same observations (i.e. sampling without replacement), use `size` to get bootstrapped data with a specific number of observations. However, specifying the `size`-argument is much less memory-efficient than the bootstrap with replacement. Hence, it is recommended to ignore the `size`-argument, if it is not really needed.

Value

A `data_frame` with one column: a list-variable `strap`, which contains the bootstrapped samples from data.

Note

This function applies nonparametric bootstrapping, i.e. the function draws samples with replacement.

See Also

[boot_ci](#) to calculate confidence intervals from bootstrap samples.

Examples

```
data(efc)
bs <- bootstrap(efc, 5)

# now run models for each bootstrapped sample
lapply(bs$strap, function(x) lm(neg_c_7 ~ e42dep + c161sex, data = x))

# generate bootstrap samples with 600 observations for each sample
bs <- bootstrap(efc, 5, 600)

# generate bootstrap samples with 70% observations of the original sample size
bs <- bootstrap(efc, 5, .7)
```

boot_ci

Standard Error and Confidence Intervals for bootstrapped estimates

Description

Compute nonparametric bootstrap standard error, confidence intervals and p-value for a vector of bootstrap replicate estimates.

Usage

```
boot_ci(data, ...)
```

```
boot_se(data, ...)
```

```
boot_p(data, ...)
```

Arguments

data	A data frame that contains the vector with bootstrapped estimates, or directly the vector (see 'Examples').
...	Optional, names of the variables with bootstrapped estimates. Required, if either data is a data frame and no vector, or if only selected variables from data should be used in the function.

Details

This method requires one or more vectors of bootstrap replicate estimates as input. The function then computes the nonparametric bootstrap standard error by calculating the standard deviation of the input vector. The mean value of the input vector and its standard error is used to calculate the lower and upper confidence interval, assuming a t-distribution of bootstrap estimate replicates. P-values from boot_p are also based on t-statistics, assuming normal distribution.

Value

A `data_frame` with either bootstrap standard error, the lower and upper confidence intervals or the p-value for all bootstrapped estimates.

See Also

`bootstrap` to generate nonparametric bootstrap samples.

Examples

```
data(efc)
bs <- bootstrap(efc, 100)

# now run models for each bootstrapped sample
bs$models <- lapply(bs$strap, function(x) lm(neg_c_7 ~ e42dep + c161sex, data = x))

# extract coefficient "dependency" and "gender" from each model
bs$dependency <- unlist(lapply(bs$models, function(x) coef(x)[2]))
bs$gender <- unlist(lapply(bs$models, function(x) coef(x)[3]))

# get bootstrapped confidence intervals
boot_ci(bs$dependency)

# compare with model fit
fit <- lm(neg_c_7 ~ e42dep + c161sex, data = efc)
confint(fit)[2, ]

# alternative function calls.
boot_ci(bs$dependency)
boot_ci(bs, dependency)
boot_ci(bs, dependency, gender)

# compare coefficients
mean(bs$dependency)
coef(fit)[2]

# bootstrap() and boot_ci() work fine within pipe-chains
library(dplyr)
efc %>%
  bootstrap(100) %>%
  mutate(models = lapply(.$strap, function(x) {
    lm(neg_c_7 ~ e42dep + c161sex, data = x)
  })) %>%
  mutate(dependency = unlist(lapply(.$models, function(x) coef(x)[2]))) %>%
  boot_ci(dependency)

# check p-value
boot_p(bs$gender)
summary(fit)$coefficients[3, ]
```

```

# 'spread_coef()' from the 'sjmisc'-package makes it easy to generate
# bootstrapped statistics like confidence intervals or p-values
library(dplyr)
library(sjmisc)
efc %>%
  # generate bootstrap replicates
  bootstrap(100) %>%
  # apply lm to all bootstrapped data sets
  mutate(models = lapply(.$strap, function(x) {
    lm(neg_c_7 ~ e42dep + c161sex + c172code, data = x)
  })) %>%
  # spread model coefficient for all 100 models
  spread_coef(models) %>%
  # compute the CI for all bootstrapped model coefficients
  boot_ci(e42dep, c161sex, c172code)

# or...
efc %>%
  # generate bootstrap replicates
  bootstrap(100) %>%
  # apply lm to all bootstrapped data sets
  mutate(models = lapply(strap, function(x) {
    lm(neg_c_7 ~ e42dep + c161sex + c172code, data = x)
  })) %>%
  # spread model coefficient for all 100 models
  spread_coef(models, append = FALSE) %>%
  # compute the CI for all bootstrapped model coefficients
  boot_ci()

```

chisq_gof

Chi-square goodness-of-fit-test

Description

This method performs a Chi-square goodness-of-fit-test (GOF) either on a numeric vector against probabilities, or a Goodness-of-fit test for [glm](#)s for binary data.

Usage

```
chisq_gof(x, prob = NULL, weights = NULL)
```

Arguments

x	Numeric vector, or a glm -object.
prob	Vector of probabilities (indicating the population probabilities) of the same length as x's amount of categories / factor levels. Use <code>nrow(table(x))</code> to determine the amount of necessary values for prob. Only used, when x is a vector, and not a glm -object.

weights Vector with weights, used to weight x.

Value

For vectors, returns the object of the computed `chisq.test`.

For `glm`-objects, an object of class `chisq_gof` with following values:

- `p.value` the p-value for the goodness-of-fit test
- `z.score` the standardized z-score for the goodness-of-fit test
- `RSS` the residual sums of squares term
- `X2` the pearson chi-squared statistic

Note

For vectors, this function is a convenient function for the `chisq.test`, performing goodness-of-fit test.

For `glm`-objects, this function performs a goodness-of-fit test based on the `X2GOF` test function of the **binomTools** package. A well-fitting model shows no significant difference between the model and the observed data, i.e. the reported p-values should be greater than 0.05.

Examples

```
data(efc)
# differing from population
chisq_gof(efc$e42dep, c(0.3,0.2,0.22,0.28))
# equal to population
chisq_gof(efc$e42dep, prop.table(table(efc$e42dep)))

# goodness-of-fit test for logistic regression
efc$services <- ifelse(efc$tot_sc_e > 0, 1, 0)
fit <- glm(services ~ neg_c_7 + c161sex + e42dep, data = efc,
          family = binomial(link = "logit"))
chisq_gof(fit)
```

cod

Tjur's Coefficient of Discrimination

Description

This method calculates the Coefficient of Discrimination D for generalized linear (mixed) models for binary data. It is an alternative to other Pseudo-R-squared values like Nakelkerke's R2 or Cox-Snell R2.

Usage

```
cod(x)
```


Arguments

x Fitted [glm](#) or [glmer](#) model.

Value

The D Coefficient of Discrimination, also known as Tjur's R-squared value.

Note

The Coefficient of Discrimination D can be read like any other (Pseudo-)R-squared 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

See Also

[r2](#) for Nagelkerke's and Cox and Snell's pseudo r-squared coefficients.

Examples

```
library(sjmisc)
data(efc)

# Tjur's R-squared value
efc$services <- ifelse(efc$tot_sc_e > 0, 1, 0)
fit <- glm(services ~ neg_c_7 + c161sex + e42dep,
          data = efc, family = binomial(link = "logit"))
cod(fit)
```

converge_ok

Convergence test for mixed effects models

Description

This function provides an alternative convergence test for [merMod](#)-objects.

Usage

```
converge_ok(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

This function provides an alternative convergence test for `merMod`-objects, as discussed [here](#) and suggested by Ben Bolker in [this comment](#).

Value

Logical vector, TRUE if convergence is fine, FALSE if convergence is suspicious. Additionally, the convergence value is returned as return value's name.

Examples

```
library(sjmisc)
library(lme4)
data(efc)
# create binary response
efc$hi_qol <- dicho(efc$quol_5)
# prepare group variable
efc$grp = as.factor(efc$e15relat)
# data frame for fitted model
mydf <- data.frame(hi_qol = as.factor(efc$hi_qol),
                  sex = as.factor(efc$c161sex),
                  c12hour = as.numeric(efc$c12hour),
                  neg_c_7 = as.numeric(efc$neg_c_7),
                  grp = efc$grp)

# fit glmer
fit <- glmer(hi_qol ~ sex + c12hour + neg_c_7 + (1|grp),
            data = mydf, family = binomial("logit"))

converge_ok(fit)
```

cramer

Cramer's V for a contingency table

Description

Compute Cramer's V for a table with more than 2x2 fields.

Usage

```
cramer(tab)
```

Arguments

`tab` A `table` or `fTable`. Tables of class `xTable` and other will be coerced to `fTable` objects.

Value

The table's Cramer's V.

See Also

[phi](#)

Examples

```
tab <- table(sample(1:2, 30, TRUE), sample(1:3, 30, TRUE))
cramer(tab)
```

cronb

Cronbach's Alpha for a matrix or data frame

Description

This function calculates the Cronbach's alpha value of a data frame or matrix.

Usage

```
cronb(data)
```

Arguments

data data.frame or matrix with more than 2 columns.

Value

The Cronbach's alpha value for data.

Note

See 'Examples' from [sjp.pca](#) and [sjt.pca](#).

See Also

[reliab_test](#)

`cv`*Coefficient of Variation*

Description

Compute coefficient of variation for single variables (standard deviation divided by mean) or for fitted linear (mixed effects) models (root mean squared error (RMSE) divided by mean of dependent variable).

Usage

```
cv(x, ...)
```

Arguments

`x` (Numeric) vector or a fitted linear model of class `lm`, `merMod` (**lme4**) or `lme` (**nlme**).

`...` More fitted model objects, to compute multiple coefficients of variation at once.

Details

The advantage of the `cv` is that it is unitless. This allows coefficient of variation to be compared to each other in ways that other measures, like standard deviations or root mean squared residuals, cannot be.

“It is interesting to note the differences between a model’s CV and R-squared values. Both are unitless measures that are indicative of model fit, but they define model fit in two different ways: CV evaluates the relative closeness of the predictions to the actual values while R-squared evaluates how much of the variability in the actual values is explained by the model.” (*source: UCLA-FAQ*)

Value

The coefficient of variation of `x`.

References

- [UCLA-FAQ: What is the coefficient of variation?](#)
- Everitt, Brian (1998). The Cambridge Dictionary of Statistics. Cambridge, UK New York: Cambridge University Press

See Also

[rmse](#)

Examples

```
data(efc)
cv(efc$e17age)

fit <- lm(neg_c_7 ~ e42dep, data = efc)
cv(fit)

library(lme4)
fit <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
cv(fit)

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

deff

Design effects for two-level mixed models

Description

Compute the design effect (also called *Variance Inflation Factor*) for mixed models with two-level design.

Usage

```
deff(n, icc = 0.05)
```

Arguments

n	Average number of observations per grouping cluster (i.e. level-2 unit).
icc	Assumed intraclass correlation coefficient for multilevel-model.

Value

The design effect (Variance Inflation Factor) for the two-level model.

References

- Bland JM. 2000. Sample size in guidelines trials. *Fam Pract.* (17), 17-20.
- Hsieh FY, Lavori PW, Cohen HJ, Feussner JR. 2003. An Overview of Variance Inflation Factors for Sample-Size Calculation. *Evaluation & the Health Professions* 26: 239–257. doi: [10.1177/0163278703255230](https://doi.org/10.1177/0163278703255230)
- Snijders TAB. 2005. Power and Sample Size in Multilevel Linear Models. In: Everitt BS, Howell DC (Hrsg.). *Encyclopedia of Statistics in Behavioral Science*. Chichester, UK: John Wiley & Sons, Ltd. doi: [10.1002/0470013192.bsa492](https://doi.org/10.1002/0470013192.bsa492)
- Thompson DM, Fernald DH, Mold JW. 2012. Intraclass Correlation Coefficients Typical of Cluster-Randomized Studies: Estimates From the Robert Wood Johnson Prescription for Health Projects. *The Annals of Family Medicine*;10(3):235–40. doi: [10.1370/afm.1347](https://doi.org/10.1370/afm.1347)

Examples

```
# Design effect for two-level model with 30 cluster groups
# and an assumed intraclass correlation coefficient of 0.05.
deff(n = 30)

# Design effect for two-level model with 24 cluster groups
# and an assumed intraclass correlation coefficient of 0.2.
deff(n = 24, icc = 0.2)
```

efc

Sample dataset from the EUROFAMCARE project

Description

A SPSS sample data set, read with the [read_spss](#) function and "converted" with [unlabel](#).

Examples

```
# Attach EFC-data
data(efc)

# Show structure
str(efc)

# show first rows
head(efc)

# show variables
## Not run:
library(sjmisc)
library(sjPlot)
view_df(efc)

# show variable labels
get_label(efc)

# plot efc-data frame summary
sjt.df(efc, alternateRowColor = TRUE)
## End(Not run)
```

eta_sq	<i>Eta-squared of fitted anova</i>
--------	------------------------------------

Description

Returns the eta-squared value for one-way-anovas.

Usage

```
eta_sq(...)
```

Arguments

... Fitted one-way-anova model or a dependent and grouping variable (see 'Examples').

Value

The eta-squared value.

Note

Interpret eta-squared like r-squared or R-squared; a rule of thumb (Cohen):

- .02 ~ small
- .13 ~ medium
- .26 ~ large

References

- [How to compute eta-sq in ANOVA by hand?](#)
- [How to interpret and report eta squared?](#)
- [Wikipedia: Eta-squared](#)
- Levine TR, Hullett CR (2002): Eta Squared, Partial Eta Squared, and Misreporting of Effect Size in Communication Research ([pdf](#))

Examples

```
# load sample data
data(efc)

# fit linear model
fit <- aov(c12hour ~ as.factor(e42dep), data = efc)

# print eta squared
eta_sq(fit)
```

```
# grouping variable will be converted to factor autoamtically
eta_sq(efc$c12hour, efc$e42dep)
```

get_model_pval *Get p-values from regression model objects*

Description

This function returns the p-values for fitted model objects.

Usage

```
get_model_pval(fit, p.kr = FALSE)
```

Arguments

fit	A fitted model object of lm, glm, merMod, merModLmerTest, pggls or gls. Other classes may work as well.
p.kr	Logical, if TRUE, the computation of p-values is based on conditional F-tests with Kenward-Roger approximation for the df (see 'Details').

Details

For linear mixed models (lmerMod-objects), see [merMod_p](#).

Value

A [tibble](#) with the model coefficients' names (term), p-values (p.value) and standard errors (std.error).

Examples

```
data(efc)
# linear model fit
fit <- lm(neg_c_7 ~ e42dep + c172code, data = efc)
get_model_pval(fit)

# Generalized Least Squares fit
library(nlme)
fit <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
          correlation = corAR1(form = ~ 1 | Mare))
get_model_pval(fit)

# lme4-fit
library(lme4)
fit <- lmer(Reaction ~ Days + (Days | Subject), data = sleepstudy)
get_model_pval(fit, p.kr = TRUE)
```

get_re_var	<i>Random effect variances</i>
------------	--------------------------------

Description

This function extracts specific components from the random effect variances or the random-intercept-slope-correlation of mixed effects models. Currently, only [merMod](#) objects are supported.

Usage

```
get_re_var(x, comp = c("tau.00", "tau.01", "tau.11", "rho.01", "sigma_2"))
```

Arguments

x	An object of class <code>icc.lme4</code> , as returned by the icc function, or a fitted mixed effects model (merMod -class).
comp	Name of the variance component to be returned. See 'Details'.

Details

The random effect variances indicate the between- and within-group variances as well as random-slope variance and random-slope-intercept correlation. Use following values for `comp` to get the particular variance component:

- "sigma_2" Within-group (residual) variance
- "tau.00" Between-group-variance (variation between individual intercepts and average intercept)
- "tau.11" Random-slope-variance (variation between individual slopes and average slope)
- "tau.01" Random-Intercept-Slope-covariance
- "rho.01" Random-Intercept-Slope-correlation

Value

The required variance component.

See Also

[icc](#) and [re_var](#)

Examples

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

# icc
icc(fit)

# all random effect variance components
```

```
re_var(fit)

# just the rand. slope-intercept covariance
get_re_var(fit, "tau.01")

sleepstudy$mygrp <- sample(1:45, size = 180, replace = TRUE)
fit <- lmer(Reaction ~ Days + (1 | mygrp) + (Days | Subject), sleepstudy)
get_re_var(fit, "rho.01")
```

hoslem_gof

Hosmer-Lemeshow Goodness-of-fit-test

Description

This method performs a Hosmer-Lemeshow goodness-of-fit-test for generalized linear (mixed) models for binary data.

Usage

```
hoslem_gof(x, g = 10)
```

Arguments

x	Fitted glm or glmer model.
g	Number of bins to divide the data. Default is 10.

Value

An object of class `hoslem_test` with following values:

- `chisq` the Hosmer-Lemeshow chi-squared statistic
- `df` degrees of freedom
- `p.value` the p-value for the goodness-of-fit test

Note

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.

See Also

[r2](#)

Examples

```
data(efc)
# goodness-of-fit test for logistic regression
efc$services <- ifelse(efc$tot_sc_e > 0, 1, 0)
fit <- glm(services ~ neg_c_7 + c161sex + e42dep, data = efc,
          family = binomial(link = "logit"))
hoslem_gof(fit)
```

 icc

Intraclass-Correlation Coefficient

Description

This function calculates the intraclass-correlation (icc) for random intercepts of mixed effects models. Currently, only [merMod](#) objects are supported.

Usage

```
icc(x, ...)
```

Arguments

x	Fitted mixed effects model (merMod -class).
...	More fitted model objects, to compute multiple intraclass-correlation coefficients at once.

Details

The calculation of the ICC for generalized linear mixed models with binary outcome is based on [Wu et al. \(2012\)](#). For Poisson multilevel models, please refer to [Stryhn et al. \(2006\)](#). [Aly et al. \(2014\)](#) describe computation of ICC for negative binomial models.

There is a `print`-method that prints the variance parameters using the `comp`-argument set to `"var"`: `print(x, comp = "var")` (see 'Examples'). The `re_var`-function is a convenient wrapper.

The random effect variances indicate the between- and within-group variances as well as random-slope variance and random-slope-intercept correlation. The components are denoted as following:

- Within-group (residual) variance: `sigma_2`
- Between-group-variance: `tau.00` (variation between individual intercepts and average intercept)
- Random-slope-variance: `tau.11` (variation between individual slopes and average slope)
- Random-Intercept-Slope-covariance: `tau.01`
- Random-Intercept-Slope-correlation: `rho.01`

Value

A numeric vector with all random intercept intraclass-correlation-coefficients, or a list of numeric vectors, when more than one model were used as arguments. Furthermore, between- and within-group variances as well as random-slope variance are returned as attributes.

Note

Some notes on why the ICC is useful, based on *Grace-Martin*:

- It can help you determine whether or not a linear mixed model is even necessary. If you find that the correlation is zero, that means the observations within clusters are no more similar than observations from different clusters. Go ahead and use a simpler analysis technique.
- It can be theoretically meaningful to understand how much of the overall variation in the response is explained simply by clustering. For example, in a repeated measures psychological study you can tell to what extent mood is a trait (varies among people, but not within a person on different occasions) or state (varies little on average among people, but varies a lot across occasions).
- It can also be meaningful to see how the ICC (as well as the between and within cluster variances) changes as variable are added to the model.

In short, the ICC can be interpreted as “the proportion of the variance explained by the grouping structure in the population” (*Hox 2002: 15*).

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 a level-2 variable explains the portion of variation in the grouping structure (random intercept).

Caution: For three-level-models, depending on the nested structure of the model, the ICC only reports the proportion of variance explained for each grouping level. However, 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 computed manually. Use `get_re_var` to get the 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(get_re_var(fit)) / (sum(get_re_var(fit)) + get_re_var(fit, "sigma_2"))
```

or for the ICC between level 2 and 3:

```
get_re_var(fit)[2] / sum(get_re_var(fit))
```

References

- Aguinis H, Gottfredson RK, Culpepper SA. 2013. Best-Practice Recommendations for Estimating Cross-Level Interaction Effects Using Multilevel Modeling. *Journal of Management* 39(6): 1490–1528 (doi: [10.1177/0149206313478188](https://doi.org/10.1177/0149206313478188))
- Aly SS, Zhao J, Li B, Jiang J. 2014. Reliability of environmental sampling culture results using the negative binomial intraclass correlation coefficient. *Springerplus* [Internet] 3. Available from: <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3916583/>

- Grace-Martion K. The Intraclass Correlation Coefficient in Mixed Models, [web](#)
- Hox J. 2002. Multilevel analysis: techniques and applications. Mahwah, NJ: Erlbaum
- Rabe-Hesketh S, Skrondal A. 2012. Multilevel and longitudinal modeling using Stata. 3rd ed. College Station, Tex: Stata Press Publication
- Raudenbush SW, Bryk AS. 2002. Hierarchical linear models: applications and data analysis methods. 2nd ed. Thousand Oaks: Sage Publications
- Stryhn H, Sanchez J, Morley P, Booker C, Dohoo IR. 2006. Interpretation of variance parameters in multilevel Poisson regression models. Proceedings of the 11th International Symposium on Veterinary Epidemiology and Economics, 2006 Available at <http://www.sciquest.org.nz/node/64294>
- Wu S, Crespi CM, Wong WK. 2012. Comparison of methods for estimating the intraclass correlation coefficient for binary responses in cancer prevention cluster randomized trials. *Contemporary Clinical Trials* 33: 869-880 (doi: [10.1016/j.cct.2012.05.004](https://doi.org/10.1016/j.cct.2012.05.004))

Further helpful online-resources:

- [CrossValidated \(2012\) Intraclass correlation \(ICC\) for an interaction?](#)
- [CrossValidated \(2014\) Interpreting the random effect in a mixed-effect model](#)
- [CrossValidated \(2014\) how to partition the variance explained at group level and individual level](#)

See Also

`re_var` to print random effect variances and `get_re_var` to get the values from a particular variance component.

Examples

```
library(lme4)
fit0 <- lmer(Reaction ~ 1 + (1 | Subject), sleepstudy)
icc(fit0)

fit1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
icc(fit1)

sleepstudy$mygrp <- sample(1:45, size = 180, replace = TRUE)
fit2 <- lmer(Reaction ~ Days + (1 | mygrp) + (Days | Subject), sleepstudy)
icc(fit2)

# return icc for all models at once
icc(fit0, fit1, fit2)

icc1 <- icc(fit1)
icc2 <- icc(fit2)

print(icc1, comp = "var")
print(icc2, comp = "var")
```

`inequ_trend`*Compute trends in status inequalities*

Description

This method computes the proportional change of absolute (rate differences) and relative (rate ratios) inequalities of prevalence rates for two different status groups, as proposed by Mackenbach et al. (2015).

Usage

```
inequ_trend(data, prev.low, prev.hi)
```

Arguments

<code>data</code>	A data frame that the variables with prevalence rates for both low and high status groups (see 'Examples').
<code>prev.low</code>	The name of the variable with the prevalence rates for the low status groups.
<code>prev.hi</code>	The name of the variable with the prevalence rates for the hi status groups.

Details

Given the time trend of prevalence rates of an outcome for two status groups (e.g. the mortality rates for people with lower and higher socioeconomic status over 40 years), this function computes the proportional change of absolute and relative inequalities, expressed in changes in rate differences and rate ratios. The function implements the algorithm proposed by *Mackenbach et al. 2015*.

Value

A data frame with the prevalence rates as well as the values for the proportional change in absolute (rd) and relative (rr) inequalities.

References

Mackenbach JP, Martikainen P, Menvielle G, de Gelder R. 2015. The Arithmetic of Reducing Relative and Absolute Inequalities in Health: A Theoretical Analysis Illustrated with European Mortality Data. *Journal of Epidemiology and Community Health* 70(7): 730–36. doi: [10.1136/jech-2015-207018](https://doi.org/10.1136/jech-2015-207018)

Examples

```
# This example reproduces Fig. 1 of Mackenbach et al. 2015, p.5  
  
# 40 simulated time points, with an initial rate ratio of 2 and  
# a rate difference of 100 (i.e. low status group starts with a  
# prevalence rate of 200, the high status group with 100)  
  
# annual decline of prevalence is 1% for the low, and 3% for the
```

```
# high status group

n <- 40
time <- seq(1, n, by = 1)
lo <- rep(200, times = n)
for (i in 2:n) lo[i] <- lo[i - 1] * .99

hi <- rep(100, times = n)
for (i in 2:n) hi[i] <- hi[i - 1] * .97

prev.data <- data.frame(lo, hi)

# print values
inequ_trend(prev.data, lo, hi)

# plot trends - here we see that the relative inequalities
# are increasing over time, while the absolute inequalities
# are first increasing as well, but later are decreasing
# (while rel. inequ. are still increasing)
plot(inequ_trend(prev.data, lo, hi))
```

levene_test

Plot Levene-Test for One-Way-Anova

Description

Plot results of Levene's Test for Equality of Variances for One-Way-Anova.

Usage

```
levene_test(depVar, grpVar)
```

Arguments

depVar	Dependent variable.
grpVar	Grouping (independent) variable, as unordered factor.

Examples

```
data(efc)
levene_test(efc$c12hour, efc$e42dep)
```

`mean_n`*Row means with min amount of valid values*

Description

This function is similar to the SPSS `MEAN.n` function and computes row means from a `data.frame` or `matrix` if at least `n` values of a row are valid (and not `NA`).

Usage

```
mean_n(dat, n, digits = 2)
```

Arguments

<code>dat</code>	<code>data.frame</code> with at least two columns, where row means are applied.
<code>n</code>	May either be <ul style="list-style-type: none">• a numeric value that indicates the amount of valid values per row to calculate the row mean;• or a value between 0 and 1, indicating a proportion of valid values per row to calculate the row mean (see 'Details'). If a row's amount of valid values is less than <code>n</code> , <code>NA</code> will be returned as row mean value.
<code>digits</code>	Numeric value indicating the number of decimal places to be used for rounding mean value. Negative values are allowed (see 'Details').

Details

Rounding to a negative number of `digits` means rounding to a power of ten, so for example `mean_n(df, 3, digits = -2)` rounds to the nearest hundred.

For `n`, must be a numeric value from 0 to `ncol(dat)`. If a *row* in `dat` has at least `n` non-missing values, the row mean is returned. If `n` is a non-integer value from 0 to 1, `n` is considered to indicate the proportion of necessary non-missing values per row. E.g., if `n = .75`, a row must have at least `ncol(dat) * n` non-missing values for the row mean to be calculated. See 'Examples'.

Value

A vector with row mean values of `df` for those rows with at least `n` valid values. Else, `NA` is returned.

References

r4stats.com

Examples

```

dat <- data.frame(c1 = c(1,2,NA,4),
                 c2 = c(NA,2,NA,5),
                 c3 = c(NA,4,NA,NA),
                 c4 = c(2,3,7,8))

# needs at least 4 non-missing values per row
mean_n(dat, 4) # 1 valid return value

# needs at least 3 non-missing values per row
mean_n(dat, 3) # 2 valid return values

# needs at least 2 non-missing values per row
mean_n(dat, 2)

# needs at least 1 non-missing value per row
mean_n(dat, 1) # all means are shown

# needs at least 50% of non-missing values per row
mean_n(dat, .5) # 3 valid return values

# needs at least 75% of non-missing values per row
mean_n(dat, .75) # 2 valid return values

```

merMod_p

*Compute p-values for merMod objects***Description**

This function computes p-values for mixed effects models (merMod-objects) that have been fitted with the **lme4**-package.

Usage

```
merMod_p(x, p.kr = TRUE)
```

Arguments

x	A fitted (generalized) linear (mixed) model (merMod-object).
p.kr	Logical, if TRUE, the computation of p-values is based on conditional F-tests with Kenward-Roger approximation for the df (see 'Details').

Details

For linear mixed models (lmerMod-objects), the computation of p-values (if p.kr = TRUE) is based on conditional F-tests with Kenward-Roger approximation for the df, using the **pbkrtest**-package. If **pbkrtest** is not available or p.kr = FALSE, or if x is a glmerMod-object, computation of p-values

is based on normal-distribution assumption, treating the t-statistics as Wald z-statistics.

If p-values already have been computed (e.g. for `merModLmerTest`)-objects from the **lmerTest**-package), these will be returned.

Value

A named vector with p-values for the model coefficients.

Examples

```
library(lme4)
# fit model
fit <- lmer(Reaction ~ Days + (Days | Subject), data = sleepstudy)
merMod_p(fit)
merMod_p(fit, p.kr = FALSE)
```

mic	<i>Mean Inter-Item-Correlation</i>
-----	------------------------------------

Description

This function calculates a mean inter-item-correlation, i.e. a correlation matrix of data will be computed (unless data 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.

Usage

```
mic(data, cor.method = c("pearson", "spearman", "kendall"))
```

Arguments

<code>data</code>	A matrix as returned by the <code>cor</code> -function, or a <code>data.frame</code> , where correlations will be calculated.
<code>cor.method</code>	Indicates the correlation computation method. May be one of "spearman" (default), "pearson" or "kendall". You may use initial letter only.

Value

The value of the computed mean inter-item-correlation.

Note

“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*)

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: [10.1007/978-94-007-0753-5_1493](https://doi.org/10.1007/978-94-007-0753-5_1493)

Examples

```
# Data from the EUROFAMCARE sample dataset
data(efc)
# receive first item of COPE-index scale
start <- which(colnames(efc) == "c82cop1")
# receive last item of COPE-index scale
end <- which(colnames(efc) == "c90cop9")
# create data frame with COPE-index scale
mydat <- data.frame(efc[, c(start:end)])

mic(mydat)
```

mwu

Mann-Whitney-U-Test

Description

This function performs a Mann-Whitney-U-Test (or Wilcoxon rank sum test, see [wilcox.test](#) and [wilcox_test](#)) for x , for each group indicated by grp . If grp has more than two categories, a comparison between each combination of two groups is performed.

The function reports U , p and Z -values as well as effect size r and group-rank-means.

Usage

```
mwu(x, grp, distribution = "asymptotic", weights = NULL)
```

Arguments

x	Numeric vector / variable, where the Mann-Whitney-U-Test should be applied to
grp	Grouping variable indicating the groups that should be used for comparison
$distribution$	Indicates how the null distribution of the test statistic should be computed. May be one of "exact", "approximate" or "asymptotic" (default). See wilcox_test for details.
$weights$	Integer valued weights for the observations. By default, this is NULL.

Value

(Invisibly) returns a data frame with U , p and Z -values for each group-comparison as well as effect-size r ; additionally, group-labels and groups' n 's are also included.

Note

This function calls the `wilcox_test` with formula. If `grp` has more than two groups, additionally a Kruskal-Wallis-Test (see `kruskal.test`) is performed.

Interpretation of effect sizes, as a rule-of-thumb:

- small effect ≥ 0.1
- medium effect ≥ 0.3
- large effect ≥ 0.5

Examples

```
data(efc)
# Mann-Whitney-U-Tests for elder's age by elder's dependency.
mwu(efc$e17age, efc$e42dep)
```

odds_to_rr

Get relative risks estimates from logistic regressions

Description

This function converts odds ratios from a logistic regression model (including mixed models) into relative risks.

Usage

```
odds_to_rr(fit)
```

Arguments

`fit` A fitted binomial generalized linear (mixed) model with logit-link function (logistic (multilevel) regression model).

Details

This function extracts the odds ratios (exponentiated model coefficients) from logistic regressions (fitted with `glm` or `glmer`) and their related confidence intervals, and transforms these values into relative risks (and their related confidence intervals).

The formula for transformation is based on Zhang and Yu (1998): $RR \leftarrow OR / ((1 - P_0) + (P_0 * OR))$, where `OR` is the odds ratio and `P0` indicates the proportion of the incidence in the outcome variable.

Value

A data frame with relative risks and lower/upper confidence interval for the relative risks estimates.

References

Zhang J, Yu KF. 1998. What's the Relative Risk? A Method of Correcting the Odds Ratio in Cohort Studies of Common Outcomes. *JAMA*; 280(19): 1690-1. doi: [10.1001/jama.280.19.1690](https://doi.org/10.1001/jama.280.19.1690)

Examples

```
library(sjmisc)
library(lme4)
# create binary response
sleepstudy$Reaction.dicho <- dichotomize(sleepstudy$Reaction, dich.by = "median")
# fit model
fit <- glmer(Reaction.dicho ~ Days + (Days | Subject),
            data = sleepstudy, family = binomial("logit"))
# convert to relative risks
odds_to_rr(fit)
```

```
data(efc)
# create binary response
y <- ifelse(efc$neg_c_7 < median(na.omit(efc$neg_c_7)), 0, 1)
# create data frame for fitted model
mydf <- data.frame(y = as.factor(y),
                  sex = efc$c161sex,
                  dep = to_factor(efc$e42dep),
                  barthel = efc$barthtot,
                  education = to_factor(efc$c172code))
# fit model
fit <- glm(y ~., data = mydf, family = binomial(link = "logit"))
# convert to relative risks
odds_to_rr(fit)
```

overdisp

Check overdispersion of GL(M)M's

Description

This function checks generalized linear (mixed) models for overdispersion.

Usage

```
overdisp(x, trafo = NULL)
```

Arguments

x	Fitted GLMM (merMod -class) or <code>glm</code> model.
trafo	A specification of the alternative, can be numeric or a (positive) function or <code>NULL</code> (the default). See 'Details' in dispersiontest in package AER . Does not apply to <code>merMod</code> objects.

Details

For merMod-objects, this function is based on the code in the [DRAFT r-sig-mixed-models FAQ](#), section *How can I deal with overdispersion in GLMMs?*. Note that this function only returns an *approximate* estimate of an overdispersion parameter.

For glm's, this function simply wraps the dispersiontest from the **AER**-package.

Value

Information on the overdispersion test.

Note

The interpretation of the returned p-value differs between GLM and GLMM. For GLMs, a p-value < .05 indicates overdispersion, while for GLMMs, a p-value > .05 indicates overdispersion.

References

[DRAFT r-sig-mixed-models FAQ](#)

Examples

```
library(sjmisc)
data(efc)

# response has many zero-counts, poisson models
# might be overdispersed
barplot(table(efc$tot_sc_e))

fit <- glm(tot_sc_e ~ neg_c_7 + e42dep + c160age,
           data = efc, family = poisson)
overdisp(fit)

library(lme4)
efc$e15relat <- to_factor(efc$e15relat)
fit <- glmer(tot_sc_e ~ neg_c_7 + e42dep + c160age + (1 | e15relat),
            data = efc, family = poisson)
overdisp(fit)
```

phi

Phi value for contingency tables

Description

Compute Phi value for a contingency table.

Usage

```
phi(tab)
```

Arguments

tab A [table](#) or [ftable](#). Tables of class [xtabs](#) and other will be coerced to [ftable](#) objects.

Value

The table's Phi value.

See Also

[cramer](#)

Examples

```
tab <- table(sample(1:2, 30, TRUE), sample(1:2, 30, TRUE))
phi(tab)
```

pred_vars

Get predictor and response variables from models

Description

pred_vars and resp_var return the names of a model's predictor or response variables as character vector.

resp_val returns the values of the model's response vector.

Usage

```
pred_vars(x)
```

```
resp_var(x)
```

```
resp_val(x)
```

Arguments

x A fitted model.

Value

The name(s) of the response or predictor variables from x as character vector; or the values from x's response vector.

Examples

```

data(efc)
fit <- lm(neg_c_7 ~ e42dep + c161sex, data = efc)

pred_vars(fit)
resp_var(fit)

resp_val(fit)

```

r2

*Compute R-squared of (generalized) linear (mixed) models***Description**

Compute R-squared values of linear (mixed) models, or pseudo-R-squared values for generalized linear (mixed) models.

Usage

```
r2(x, n = NULL)
```

Arguments

x	Fitted model of class <code>lm</code> , <code>glm</code> , <code>lmerMod/lme</code> or <code>glmerMod</code> .
n	Optional, a <code>lmerMod</code> object, representing the fitted null-model (unconditional model) to x. If n is given, the pseudo-r-squared for random intercept and random slope variances are computed (<i>Kwok et al. 2008</i>) as well as the Omega squared value (<i>Xu 2003</i>). See 'Examples' and 'Details'.

Details

For linear models, the r-squared and adjusted r-squared value is returned, as provided by the `summary-function`.

For linear mixed models, an r-squared approximation by computing the correlation between the fitted and observed values, as suggested by *Byrnes (2008)*, is returned as well as a simplified version of the Omega-squared value ($1 - (\text{residual variance} / \text{response variance})$), *Xu (2003)*, *Nakagawa, Schielzeth 2013*), unless n is specified.

If n is given, for linear mixed models pseudo r-squared measures based on the variances of random intercept (τ_{00} , between-group-variance) and random slope (τ_{11} , random-slope-variance), as well as the r-squared statistics as proposed by *Snijders and Bosker 2012* and the Omega-squared value ($1 - (\text{residual variance full model} / \text{residual variance null model})$) as suggested by *Xu (2003)* are returned.

For generalized linear models, Cox & Snell's and Nagelkerke's pseudo r-squared values are returned.

For generalized linear mixed models, the coefficient of determination as suggested by *Tjur (2009)* (see also [cod](#)).

Value

- For linear models, the r-squared and adjusted r-squared values.
- For linear mixed models, the r-squared and Omega-squared values.
- For `glm` objects, Cox & Snell's and Nagelkerke's pseudo r-squared values.
- For `glmerMod` objects, Tjur's coefficient of determination.

Note

If `n` is given, the Pseudo-R2 statistic is the proportion of explained variance in the random effect after adding co-variates or predictors to the model, or in short: the proportion of the explained variance in the random effect of the full (conditional) model `x` compared to the null (unconditional) model `n`.

The Omega-squared statistics, if `n` is given, is `1 -` the proportion of the residual variance of the full model compared to the null model's residual variance, or in short: the the proportion of the residual variation explained by the covariates.

The r-squared statistics for linear mixed models, if the unconditional model is also specified (see `n`), is the difference of the total variance of the null and full model divided by the total variance of the null model.

Alternative ways to assess the "goodness-of-fit" is to compare the ICC of the null model with the ICC of the full model (see [icc](#)).

References

- [DRAFT r-sig-mixed-models FAQ](#)
- Byrnes, J. 2008. Re: Coefficient of determination (R^2) when using `lme()` (<https://stat.ethz.ch/pipermail/r-sig-mixed-models/2008q2/000713.html>)
- Kwok OM, Underhill AT, Berry JW, Luo W, Elliott TR, Yoon M. 2008. Analyzing Longitudinal Data with Multilevel Models: An Example with Individuals Living with Lower Extremity Intra-Articular Fractures. *Rehabilitation Psychology* 53(3): 370–86. doi: [10.1037/a0012765](https://doi.org/10.1037/a0012765)
- Nakagawa S, Schielzeth H. 2013. A general and simple method for obtaining R^2 from generalized linear mixed-effects models. *Methods in Ecology and Evolution*, 4(2):133–142. doi: [10.1111/j.2041-210x.2012.00261.x](https://doi.org/10.1111/j.2041-210x.2012.00261.x)
- Rabe-Hesketh S, Skrondal A. 2012. *Multilevel and longitudinal modeling using Stata*. 3rd ed. College Station, Tex: Stata Press Publication
- Raudenbush SW, Bryk AS. 2002. *Hierarchical linear models: applications and data analysis methods*. 2nd ed. Thousand Oaks: Sage Publications

- Snijders TAB, Bosker RJ. 2012. Multilevel analysis: an introduction to basic and advanced multilevel modeling. 2nd ed. Los Angeles: Sage
- Xu, R. 2003. Measuring explained variation in linear mixed effects models. *Statist. Med.* 22:3527-3541. doi: [10.1002/sim.1572](https://doi.org/10.1002/sim.1572)
- 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

```
library(sjmisc)
library(lme4)
fit <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
r2(fit)

data(efc)
fit <- lm(barthtot ~ c160age + c12hour, data = efc)
r2(fit)

# Pseudo-R-squared values
efc$services <- ifelse(efc$tot_sc_e > 0, 1, 0)
fit <- glm(services ~ neg_c_7 + c161sex + e42dep,
          data = efc, family = binomial(link = "logit"))
r2(fit)

# Pseudo-R-squared values for random effect variances
fit <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
fit.null <- lmer(Reaction ~ 1 + (Days | Subject), sleepstudy)
r2(fit, fit.null)
```

reliab_test

Performs a reliability test on an item scale

Description

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.

Usage

```
reliab_test(x, scale.items = FALSE, digits = 3)
```

Arguments

x	data.frame with items (from a scale).
scale.items	Logical, if TRUE, the data frame's vectors will be scaled. Recommended, when the variables have different measures / scales.
digits	Amount of digits for Cronbach's Alpha and correlation values in returned data frame.

re_var	<i>Random effect variances</i>
--------	--------------------------------

Description

This function extracts the random effect variances as well as the random-intercept-slope-correlation of mixed effects models. Currently, only `merMod` objects are supported.

Usage

```
re_var(x)
```

Arguments

x Fitted mixed effects model (`merMod`-class).

Value

NULL, as this is just a convenient print-wrapper for the variance components returned by the `icc` function.

References

Aguinis H, Gottfredson RK, Culpepper SA. 2013. Best-Practice Recommendations for Estimating Cross-Level Interaction Effects Using Multilevel Modeling. *Journal of Management* 39(6): 1490–1528 (doi: [10.1177/0149206313478188](https://doi.org/10.1177/0149206313478188))

See Also

[icc](#) and [get_re_var](#)

Examples

```
library(lme4)
fit1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
re_var(fit1)

sleepstudy$mygrp <- sample(1:45, size = 180, replace = TRUE)
fit2 <- lmer(Reaction ~ Days + (1 | mygrp) + (Days | Subject), sleepstudy)
re_var(fit2)
```

rmse	<i>Root Mean Squared Error (RMSE)</i>
------	---------------------------------------

Description

Compute root mean squared error of fitted linear (mixed effects) models.

Usage

```
rmse(fit, normalized = FALSE)
```

Arguments

fit	Fitted linear model of class <code>lm</code> , <code>merMod</code> (<code>lme4</code>) or <code>lme</code> (<code>nlme</code>).
normalized	Logical, use TRUE if normalized rmse should be returned.

Value

The root mean squared error of `fit`; or the normalized root mean squared error of `fit` if `normalized = TRUE`.

Note

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). "RMSE can be interpreted as the standard deviation of the unexplained variance, and has the useful property of being in the same units as the response variable. Lower values of RMSE indicate better fit. RMSE is a good measure of how accurately the model predicts the response, and is the most important criterion for fit if the main purpose of the model is prediction." (*Grace-Martin K: Assessing the Fit of Regression Models*)

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

References

- [Wikipedia: RMSD](#)
- [Grace-Martin K: Assessing the Fit of Regression Models](#)

See Also

`cv` for the coefficient of variation, and `rse` for the residual standard error.

Examples

```
data(efc)
fit <- lm(barthtot ~ c160age + c12hour, data = efc)
rmse(fit)

library(lme4)
fit <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
rmse(fit)

# normalized RMSE
library(nlme)
fit <- lme(distance ~ age, data = Orthodont)
rmse(fit, normalized = TRUE)
```

rse

Residual Standard Error (RSE)

Description

Compute the residual standard error of fitted linear (mixed effects) models.

Usage

```
rse(fit)
```

Arguments

`fit` Fitted linear model of class `lm` or `merMod` (**lme4**).

Value

The residual standard error of `fit`.

Note

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

See Also

`cv` for the coefficient of variation, and `rmse` for the root mean squared error.

Examples

```

data(efc)
fit <- lm(barthtot ~ c160age + c12hour, data = efc)
rse(fit)

library(lme4)
fit <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
rse(fit)

```

se

*Standard Error for variables or coefficients***Description**

Compute standard error for a variable, for all variables of a data frame, for joint random and fixed effects coefficients of mixed models, or for intraclass correlation coefficients (ICC).

Usage

```
se(x, nsim = 100)
```

Arguments

x	(Numeric) vector, a data frame or a merMod-object as returned by the <code>lmer</code> -method.
nsim	Numeric, the number of simulations for calculating the standard error for intraclass correlation coefficients, as obtained by the <code>icc</code> -function.

Details

Unlike `se.coef`, which returns the standard error for fixed and random effects separately, this function computes the standard errors for joint (sums of) random and fixed effects coefficients. Hence, `se` returns the appropriate standard errors for `coef.merMod`.

The standard error for the `icc` is based on bootstrapping, thus, the `nsim`-argument is required. See 'Examples'.

Value

The standard error of `x`, or for each variable if `x` is a data frame, or for the coefficients of a mixed model (see `coef.merMod`).

Note

Computation of standard errors for coefficients of mixed models is based [on this code](#).

Examples

```

se(rnorm(n = 100, mean = 3))

data(efc)
se(efc[, 1:3])

library(lme4)
fit <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
se(fit)

## Not run:
# compute standard error of ICC for the linear mixed model
icc(fit)
se(icc(fit))

# the standard error for the ICC can be computed manually in this way,
# taking the fitted model example from above
library(dplyr)
dummy <- sleepstudy %>%
  # generate 100 bootstrap replicates of dataset
  bootstrap(100) %>%
  # run mixed effects regression on each bootstrap replicate
  mutate(models = lapply(.$strap, function(x) {
    lmer(Reaction ~ Days + (Days | Subject), data = x)
  })) %>%
  # compute ICC for each "bootstrapped" regression
  mutate(icc = unlist(lapply(.$models, icc)))
# now compute SE and p-values for the bootstrapped ICC, values
# may differ from above example due to random seed
boot_se(dummy, icc)
boot_p(dummy, icc)
## End(Not run)

```

se_ybar

Standard error of sample mean for mixed models

Description

Compute the standard error for the sample mean for mixed models, regarding the extent to which clustering affects the standard errors. May be used as part of the multilevel power calculation for cluster sampling (see *Gelman and Hill 2007, 447ff*).

Usage

```
se_ybar(fit)
```


Arguments

`fit` Fitted mixed effects model ([merMod](#)-class).

Value

The standard error of the sample mean of `fit`.

References

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

Examples

```
library(lme4)
fit <- lmer(Reaction ~ 1 + (1 | Subject), sleepstudy)
se_ybar(fit)
```

smpsize_lmm

Sample size for linear mixed models

Description

Compute an approximated sample size for linear mixed models (two-level-designs), based on power-calculation for standard design and adjusted for design effect for 2-level-designs.

Usage

```
smpsize_lmm(eff.size, df.n = NULL, power = 0.8, sig.level = 0.05, k,
  icc = 0.05)
```

Arguments

`eff.size` Effect size.

`df.n` Optional argument for the degrees of freedom for numerator. See 'Details'.

`power` Power of test (1 minus Type II error probability).

`sig.level` Significance level (Type I error probability).

`k` Number of cluster groups (level-2-unit) in multilevel-design.

`icc` Expected intraclass correlation coefficient for multilevel-model.

Details

The sample size calculation is based on a power-calculation for the standard design. If `df.n` is not specified, a power-calculation for an unpaired two-sample t-test will be computed (using `pwr.t.test` of the **pwr**-package). If `df.n` is given, a power-calculation for general linear models will be computed (using `pwr.f2.test` of the **pwr**-package). The sample size of the standard design is then adjusted for the design effect of two-level-designs (see `deff`). Thus, the sample size calculation is appropriate in particular for two-level-designs (see *Snijders 2005*). Models that additionally include repeated measures (three-level-designs) may work as well, however, the computed sample size may be less accurate.

Value

A list with two values: The number of subjects per cluster, and the total sample size for the linear mixed model.

References

- Cohen J. 1988. Statistical power analysis for the behavioral sciences (2nd ed.). Hillsdale,NJ: Lawrence Erlbaum.
- Hsieh FY, Lavori PW, Cohen HJ, Feussner JR. 2003. An Overview of Variance Inflation Factors for Sample-Size Calculation. *Evaluation & the Health Professions* 26: 239–257. doi: [10.1177/0163278703255230](https://doi.org/10.1177/0163278703255230)
- Snijders TAB. 2005. Power and Sample Size in Multilevel Linear Models. In: Everitt BS, Howell DC (Hrsg.). *Encyclopedia of Statistics in Behavioral Science*. Chichester, UK: John Wiley & Sons, Ltd. doi: [10.1002/0470013192.bsa492](https://doi.org/10.1002/0470013192.bsa492)

Examples

```
# Sample size for multilevel model with 30 cluster groups and a small to
# medium effect size (Cohen's d) of 0.3. 29 subjects per cluster and
# hence a total sample size of about 859 observations is needed.
smpsize_lmm(eff.size = .3, k = 30)

# Sample size for multilevel model with 20 cluster groups and a medium
# to large effect size for linear models of 0.2. Nine subjects per cluster and
# hence a total sample size of about 172 observations is needed.
smpsize_lmm(eff.size = .2, df.n = 5, k = 20, power = .9)
```

std

Standardize and center variables

Description

std computes a z-transformation (standardized and centered) on the input. center centers the input.

Usage

```
std(data, ..., include.fac = TRUE, append = FALSE)
```

```
center(data, ..., include.fac = TRUE, append = FALSE)
```

Arguments

data	A variable that should be standardized or centered, or a data frame with such variables.
...	Optional, names of the variables with that should be standardized or centered. Required, if either data is a data frame and no vector, and only selected variables from data should be used in the function.
include.fac	Logical, if TRUE, factors will be converted to numeric vectors and also standardized or centered.
append	Logical, if TRUE and data is a data frame, the standardized or centered variables will be appended as new columns to data. Variable (column) names will be the same, however, standardized variables get the suffix "_z", while centered variables get the suffix "_c".

Value

Either a vector with standardized or centered variables, if data was a vector; or a [tibble](#) with standardized or centered variables, if data was a data frame. If `append = TRUE`, standardized and centered variables are added as new columns to data.

Examples

```
library(dplyr)
data(efc)
std(efc$c160age) %>% head()
std(efc, e17age, c160age) %>% head()
std(efc, e17age, c160age, append = TRUE) %>% head()

center(efc$c160age) %>% head()
center(efc, e17age, c160age) %>% head()
center(efc, e17age, c160age, append = TRUE) %>% head()
```

std_beta

Standardized Beta coefficients and CI of lm and mixed models

Description

Returns the standardized beta coefficients, std. error and confidence intervals of a fitted linear (mixed) models, i.e. `fit` must either be of class `lm` or [merMod](#).

Usage

```
std_beta(fit, type = "std")
```

Arguments

fit	Fitted linear (mixed) model of class <code>lm</code> or <code>merMod</code> (lme4 package).
type	If <code>fit</code> is of class <code>lm</code> , normal standardized coefficients are computed by default. Use <code>type = "std2"</code> to follow Gelman's (2008) suggestion, rescaling the estimates by dividing them by two standard deviations, so resulting coefficients are directly comparable for untransformed binary predictors.

Details

“Standardized coefficients refer to how many standard deviations a dependent variable will change, per standard deviation increase in the predictor variable. Standardization of the coefficient is usually done to answer the question of which of the independent variables have a greater effect on the dependent variable in a multiple regression analysis, when the variables are measured in different units of measurement (for example, income measured in dollars and family size measured in number of individuals).” (*Source: Wikipedia*)

Value

A tibble with term names, standardized beta coefficients, standard error and confidence intervals of fit.

Note

For `gls`-objects, standardized beta coefficients may be wrong for categorical variables (factors), because the `model.matrix` for `gls` objects returns the original data of the categorical vector, and not the 'dummy' coded vectors as for other classes. See, as example:

```
head(model.matrix(lm(neg_c_7 ~ as.factor(e42dep), data = efc, na.action = na.omit)))
```

and

```
head(model.matrix(nlme::gls(neg_c_7 ~ as.factor(e42dep), data = efc, na.action = na.omit))).
```

In such cases, use `to_dummy` to create dummies from factors.

References

- [Wikipedia: Standardized coefficient](#)
- Gelman A. 2008. Scaling regression inputs by dividing by two standard deviations. *Statistics in Medicine* 27: 2865–2873. <http://www.stat.columbia.edu/~gelman/research/published/standardizing7.pdf>

Examples

```
# fit linear model
fit <- lm(Ozone ~ Wind + Temp + Solar.R, data = airquality)
# print std. beta coefficients
std_beta(fit)

# print std. beta coefficients and ci, using
# 2 sd and center binary predictors
std_beta(fit, type = "std2")

# std. beta for mixed models
library(lme4)
fit1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
std_beta(fit)
```

table_values	<i>Expected and relative table values</i>
--------------	---

Description

This function calculates a table's cell, row and column percentages as well as expected values and returns all results as lists of tables.

Usage

```
table_values(tab, digits = 2)
```

Arguments

tab	Simple table or ftable of which cell, row and column percentages as well as expected values are calculated. Tables of class xtabs and other will be coerced to ftable objects.
digits	Amount of digits for the table percentage values.

Value

(Invisibly) returns a list with four tables:

1. cell a table with cell percentages of tab
2. row a table with row percentages of tab
3. col a table with column percentages of tab
4. expected a table with expected values of tab

Examples

```
tab <- table(sample(1:2, 30, TRUE), sample(1:3, 30, TRUE))
# show expected values
table_values(tab)$expected
# show cell percentages
table_values(tab)$cell
```

weight

Weight a variable

Description

This function weights the variable `x` by a specific vector of weights.

Usage

```
weight(x, weights, digits = 0)
```

Arguments

<code>x</code>	(Unweighted) variable
<code>weights</code>	Vector with same length as <code>x</code> , which contains weight factors. Each value of <code>x</code> has a specific assigned weight in <code>weights</code> .
<code>digits</code>	Numeric value indicating the number of decimal places to be used for rounding the weighted values. By default, this value is 0, i.e. the returned values are integer values.

Value

The weighted `x`.

Note

The values of the returned vector are in sorted order, whereas the values' order of the original `x` may be spread randomly. Hence, `x` can't be used, for instance, for further cross tabulation. In case you want to have weighted contingency tables or (grouped) box plots etc., use the `weightBy` argument of most functions.

See Also

[weight2](#)

Examples

```
v <- sample(1:4, 20, TRUE)
table(v)
w <- abs(rnorm(20))
table(weight(v, w))

set.seed(1)
x <- sample(letters[1:5], size = 20, replace = TRUE)
w <- runif(n = 20)

table(x)
table(weight(x, w))
```

weight2

Weight a variable

Description

This function weights the variable `x` by a specific vector of weights. It's an alternative weight calculation to `weight`, though `weight` usage is recommended.

Usage

```
weight2(x, weights)
```

Arguments

<code>x</code>	(Unweighted) variable
<code>weights</code>	Vector with same length as <code>x</code> , which contains weight factors. Each value of <code>x</code> has a specific assigned weight in <code>weights</code> .

Details

This function sums up all weights values of the associated categories of `x`, whereas the `weight` function uses a `xtabs` formula to weight cases. Thus, this function may return a vector of different length than `x`.

Value

The weighted `x`.

Note

See 'Note' in `weight`

See Also

`weight`

Examples

```
v <- sample(1:4, 20, TRUE)
table(v)
w <- abs(rnorm(20))
table(weight2(v, w))
```

wtd_sd

Weighted standard deviation for variables

Description

Compute weighted standard deviation for a variable or for all variables of a data frame.

Usage

```
wtd_sd(x, weights = NULL)
```

Arguments

`x` (Numeric) vector or a data frame.
`weights` Numeric vector of weights.

Value

The weighted standard deviation of `x`, or for each variable if `x` is a data frame.

Examples

```
wtd_sd(rnorm(n = 100, mean = 3),
       runif(n = 100))

data(efc)
wtd_sd(efc[, 1:3], runif(n = nrow(efc)))
```

wtd_se	<i>Weighted standard error for variables</i>
--------	--

Description

Compute weighted standard error for a variable or for all variables of a data frame.

Usage

```
wtd_se(x, weights = NULL)
```

Arguments

`x` (Numeric) vector or a data frame.
`weights` Numeric vector of weights.

Value

The weighted standard error of `x`, or for each variable if `x` is a data frame.

Examples

```
wtd_se(rnorm(n = 100, mean = 3),  
       runif(n = 100))  
  
data(efc)  
wtd_se(efc[, 1:3], runif(n = nrow(efc)))
```

Index

*Topic **data**

- efc, 14

- boot_ci, 4, 5
- boot_p (boot_ci), 5
- boot_se (boot_ci), 5
- bootstrap, 4, 6

- center (std), 42
- chisq.test, 8
- chisq_gof, 7
- cod, 8, 33
- coef.merMod, 39
- converge_ok, 9
- cor, 26
- cramer, 10, 31
- cronb, 11, 35
- cv, 12, 37, 38

- data.frame, 24
- data_frame, 4, 6
- deff, 13, 42
- dispersiontest, 29

- efc, 14
- eta_sq, 15

- f table, 10, 31, 45

- get_model_pval, 16
- get_re_var, 17, 20, 21, 36
- glm, 7, 9, 18
- glmer, 9, 18
- gls, 44

- hoslem_gof, 18

- icc, 17, 19, 33, 36, 39
- inequ_trend, 22

- kruskal.test, 28

- levene_test, 23
- lm, 12, 37, 38, 44
- lme, 12, 37
- lmer, 39

- matrix, 24
- mean_n, 24
- merMod, 9, 10, 12, 17, 19, 29, 36–38, 41, 43, 44
- merMod_p, 16, 25
- mic, 26
- mwu, 27

- NA, 24

- odds_to_rr, 28
- overdisp, 29

- phi, 11, 30
- pred_vars, 31
- pwr.f2.test, 42
- pwr.t.test, 42

- r2, 9, 18, 32
- re_var, 17, 19, 21, 36
- read_spss, 14
- reliab_test, 11, 34
- resp_val (pred_vars), 31
- resp_var (pred_vars), 31
- rmse, 12, 37, 38
- rse, 37, 38

- se, 39
- se.coef, 39
- se_ybar, 40
- sjp.pca, 11
- sjstats (sjstats-package), 3
- sjstats-package, 3
- sjt.pca, 11
- smpsize_lmm, 41
- std, 42
- std_beta, 43

table, [10](#), [31](#), [45](#)
table_values, [45](#)
tibble, [16](#), [43](#)
to_dummy, [44](#)

unlabel, [14](#)

weight, [46](#), [47](#)
weight2, [46](#), [47](#)
wilcox.test, [27](#)
wilcox_test, [27](#), [28](#)
wtd_sd, [48](#)
wtd_se, [49](#)

xtabs, [10](#), [31](#), [45](#), [47](#)