Package 'afex'

August 27, 2019

Type Package

Title Analysis of Factorial Experiments

Depends R (>= 3.1.0), lme4 (>= 1.1-8)

Suggests emmeans (>= 1.4), coin, xtable, parallel, plyr, optimx, nloptr, knitr, rmarkdown, R.rsp, lattice, latticeExtra, multcomp, testthat, mlmRev, dplyr, tidyr, dfoptim, Matrix, psych, ggplot2, MEMSS, effects, carData, ggbeeswarm, nlme, cowplot, jtools, ggpubr, ggpol, MASS, glmmTMB, brms, rstanarm

Imports pbkrtest (>= 0.4-1), lmerTest (>= 3.0-0), car, reshape2, stats, methods, utils

Description Convenience functions for analyzing factorial experiments using ANOVA or mixed models. aov_ez(), aov_car(), and aov_4() allow specification of between, within (i.e., repeated-measures), or mixed (i.e., split-plot) ANOVAs for data in long format (i.e., one observation per row), automatically aggregating multiple observations per individual and cell of the design. mixed() fits mixed models using lme4::lmer() and computes p-values for all fixed effects using either Kenward-Roger or Satterthwaite approximation for degrees of freedom (LMM only), parametric bootstrap (LMMs and GLMMs), or likelihood ratio tests (LMMs and GLMMs). afex_plot() provides a high-level interface for interaction or one-way plots using ggplot2, combining raw data and model estimates. afex uses type 3 sums of squares as default (imitating commercial statistical software).

URL http://afex.singmann.science/, https://github.com/singmann/afex

BugReports https://github.com/singmann/afex/issues

License GPL (>= 2)

Encoding UTF-8

VignetteBuilder knitr, R.rsp

Version 0.25-1

RoxygenNote 6.1.1

LazyData true

NeedsCompilation no

Author Henrik Singmann [aut, cre] (<https://orcid.org/0000-0002-4842-3657>),
Ben Bolker [aut],
Jake Westfall [aut],
Frederik Aust [aut] (<https://orcid.org/0000-0003-4900-788X>),
Mattan S. Ben-Shachar [aut],
Søren Højsgaard [ctb],
John Fox [ctb],
Michael A. Lawrence [ctb],
Ulf Mertens [ctb],
Jonathon Love [ctb],
Russell Lenth [ctb],
Rune Haubo Bojesen Christensen [ctb]

Maintainer Henrik Singmann <singmann+afex@gmail.com>

Repository CRAN

Date/Publication 2019-08-27 18:10:02 UTC

R topics documented:

afex-package *Analysis of Factorial Experiments Maintainer: NA*

Description

Convenience functions for analyzing factorial experiments using ANOVA or mixed models. aov_ez(), aov_car(), and aov_4() allow specification of between, within (i.e., repeated-measures), or mixed (i.e., split-plot) ANOVAs for data in long format (i.e., one observation per row), automatically aggregating multiple observations per individual and cell of the design. mixed() fits mixed models using lme4::lmer() and computes p-values for all fixed effects using either Kenward-Roger or Satterthwaite approximation for degrees of freedom (LMM only), parametric bootstrap (LMMs and GLMMs), or likelihood ratio tests (LMMs and GLMMs). afex_plot() provides a high-level interface for interaction or one-way plots using ggplot2, combining raw data and model estimates. afex uses type 3 sums of squares as default (imitating commercial statistical software).

Details

The DESCRIPTION file:

Package:	afex
Type:	Package
Title:	Analysis of Factorial Experiments
Depends:	R (>= 3.1.0), lme4 (>= 1.1-8)
Suggests:	emmeans (>= 1.4), coin, xtable, parallel, plyr, optimx, nloptr, knitr, rmarkdown, R.rsp, lattice, latticeExtra,
Imports:	pbkrtest (>= 0.4-1), lmerTest (>= 3.0-0), car, reshape2, stats, methods, utils
Description:	Convenience functions for analyzing factorial experiments using ANOVA or mixed models. aov_ez(), aov_
URL:	http://afex.singmann.science/, https://github.com/singmann/afex
BugReports:	https://github.com/singmann/afex/issues
License:	GPL (>=2)
Encoding:	UTF-8
VignetteBuilder:	knitr, R.rsp
Authors@R:	c(person(given="Henrik", family="Singmann", role=c("aut", "cre"), email="singmann+afex@gmail.com",
Version:	0.25-1
RoxygenNote:	6.1.1
LazyData:	true
Author:	Henrik Singmann [aut, cre] (<https://orcid.org/0000-0002-4842-3657>), Ben Bolker [aut], Jake Westfall [a
Maintainer:	Henrik Singmann <singmann+afex@gmail.com>

Author(s)

Maintainer: Henrik Singmann <singmann+afex@gmail.com> (0000-0002-4842-3657)

Authors:

- Ben Bolker
- Jake Westfall

- Frederik Aust (0000-0003-4900-788X)
- Mattan S. Ben-Shachar

Other contributors:

- Søren Højsgaard [contributor]
- John Fox [contributor]
- Michael A. Lawrence [contributor]
- Ulf Mertens [contributor]
- Jonathon Love [contributor]
- Russell Lenth [contributor]
- Rune Haubo Bojesen Christensen [contributor]

See Also

Useful links:

- http://afex.singmann.science/
- https://github.com/singmann/afex
- Report bugs at https://github.com/singmann/afex/issues

`afex_aov-methods` *Methods for afex_aov objects*

Description

Methods defined for objects returned from the ANOVA functions `aov_car` et al. of class `afex_aov` containing both the ANOVA fitted via `car::Anova` and base R's `aov`.

Usage

```
## S3 method for class 'afex_aov'
anova(object, es = afex_options("es_aov"),
  observed = NULL, correction = afex_options("correction_aov"),
  MSE = TRUE, intercept = FALSE, p_adjust_method = NULL,
  sig_symbols = attr(object$anova_table, "sig_symbols"), ...)

## S3 method for class 'afex_aov'
print(x, ...)

## S3 method for class 'afex_aov'
summary(object, ...)

recover_data.afex_aov(object, ..., model = afex_options("emmeans_model"))

emm_basis.afex_aov(object, trms, xlev, grid, ...,
  model = afex_options("emmeans_model"))
```

Arguments

object, x	object of class afex_aov as returned from aov_car and related functions.
es	Effect Size to be reported. The default is given by afex_options("es_aov"), which is initially set to "ges" (i.e., reporting generalized eta-squared, see details). Also supported is partial eta-squared ("pes") or "none".
observed	character vector referring to the observed (i.e., non manipulated) variables/effects in the design. Important for calculation of generalized eta-squared (ignored if es is not "ges"), see details.
correction	Character. Which sphericity correction of the degrees of freedom should be reported for the within-subject factors. The default is given by afex_options("correction_aov"), which is initially set to "GG" corresponding to the Greenhouse-Geisser correction. Possible values are "GG", "HF" (i.e., Hyunh-Feldt correction), and "none" (i.e., no correction).
MSE	logical. Should the column containing the Mean Sqaured Error (MSE) be displayed? Default is TRUE.
intercept	logical. Should intercept (if present) be included in the ANOVA table? Default is FALSE which hides the intercept.
p_adjust_method	character indicating if p-values for individual effects should be adjusted for multiple comparisons (see p.adjust and details).
sig_symbols	Character. What should be the symbols designating significance? When entering an vector with length(sig.symbol) < 4 only those elements of the default (c(" +"," *"," **"," ***")) will be replaced. sig_symbols = "" will display the stars but not the +, sig_symbols = rep("",4) will display no symbols. The default is given by afex_options("sig_symbols").
...	further arguments passed through, see description of return value for details.
model	argument for emmeans() and rlated functions that allows to choose on which model the follow-up tests for ANOVAs with repeated-measures factors are based. "univariate" uses the aov model and "multivariate" uses the lm model. Default given by afex_options("emmeans_mode"). Multivariate tests likely provide a better correction for violations of sphericity.
trms, xlev, grid	same as for emm_basis.

Details

Exploratory ANOVA, for which no detailed hypotheses have been specified a priori, harbor a multiple comparison problem (Cramer et al., 2015). To avoid an inflation of familywise Type I error rate, results need to be corrected for multiple comparisons using p_adjust_method. p_adjust_method defaults to the method specified in the call to aov_car in anova_table. If no method was specified and p_adjust_method = NULL p-values are not adjusted.

Value

anova Returns an ANOVA table of class c("anova","data.frame"). Information such as effect size (es) or df-correction are calculated each time this method is called.

`summary` For ANOVAs containing within-subject factors it returns the full output of the within-subject tests: the uncorrected results, results containing Greenhousse-Geisser and Hyunh-Feldt correction, and the results of the Mauchly test of sphericity (all achieved via `summary.Anova.mlm`). For other ANOVAs, the `anova` table is simply returned.

`print` Prints (and invisibly returns) the ANOVA table as constructed from `nice` (i.e., as strings rounded nicely). Arguments in `...` are passed to `nice` allowing to pass arguments such as `es` and `correction`.

`recover_data` **and** `emm_basis` Provide the backbone for using `emmeans` and related functions from **emmeans** directly on `afex_aov` objects by returning a `emmGrid-class` object. Should not be called directly but through the functionality provided by **emmeans**.

References

Cramer, A. O. J., van Ravenzwaaij, D., Matzke, D., Steingroever, H., Wetzels, R., Grasman, R. P. P. P., ... Wagenmakers, E.-J. (2015). Hidden multiplicity in exploratory multiway ANOVA: Prevalence and remedies. *Psychonomic Bulletin & Review*, 1-8. doi:10.3758/s13423-015-0913-5

`afex_options` *Set/get global afex options*

Description

Global afex options are used, for example, by `aov_car` (et al.) and `mixed`. But can be changed in each functions directly using an argument (which has precedence over the global options).

Usage

```
afex_options(...)
```

Arguments

`...`	One of four: (1) nothing, then returns all options as a list; (2) a name of an option element, then returns its' value; (3) a name-value pair which sets the corresponding option to the new value (and returns nothing), (4) a list with option-value pairs which sets all the corresponding arguments. The example show all possible cases.

Details

The following arguments are currently set:

- `check_contrasts` should contrasts be checked and changed to sum-to-zero contrasts? Default is `TRUE`.
- `type` type of sums-of-squares to be used for testing effects, default is 3 which reports Type 3 tests.
- `method_mixed`: Method used to obtain p-values in `mixed`, default is `"KR"` (which will change to `"LRT"` soon). (`mixed()` only)

- es_aov: Effect size reported for ANOVAs (see aov_car), default is "ges" (generalized eta-squared).
- correction_aov: Correction used for within-subjects factors with more than two levels for ANOVAs (see aov_car or nice), default is "GG" (Greenhouse-Geisser correction). (ANOVA functions only)
- emmeans_model: Which model should be used by **emmeans** for follow-up analysis of ANOVAs (i.e., objects pf class "afex_aov")? Default is "univariate" which uses the aov model object (if present). The other option is "multivariate" which uses the lm model object (which is an object of class "mlm" in case repeated-measures factors are present).
- include_aov: Should the aov model be included into ANOVA objects of class "afex_aov"? Setting this to FALSE can lead to considerable speed improvements.
- factorize: Should between subject factors be factorized (with note) before running the analysis? Default is TRUE. (ANOVA functions only)
- sig_symbols: Default significant symbols used for ANOVA and mixed printing. Default isc(" +"," *"," **"," ***").
- lmer_function: Which lmer function should mixed or lmer_alt use. The default is "lmerTest" which uses lmer, "lme4" is also possible which uses lmer. Note that mixed methods "KR" and "S" only work with "lmerTest". For the other methods, "lme4" could be minimally faster, but does not allow to use lmerTest::anova().
- return_aov: Return value of the ANOVA functions (see aov_car), default is "nice".

Value

depends on input, see above.

Note

All options are saved in the global R options with prefix afex.

Examples

```
afex_options() # see all options

afex_options("return_aov") #get single option

aop <- afex_options() # save current options

## Not run:
# change options
afex_options(return_aov = "nice")
afex_options("return_aov") #get single option
afex_options(return_aov = "nice", method_mixed = "LRT")
afex_options("method_mixed") #get single option
# do something

## End(Not run)
afex_options(aop) # reset options
```

afex_plot *m-way Plot with Error Bars and Raw Data*

Description

Plots results from factorial experiments. Estimated marginal means and error bars are plotted in the foreground, raw data is plotted in the background. Error bars can be based on different standard errors (e.g., model-based, within-subjects, between-subjects). Functions described here return a **ggplot2** plot object, thus allowing further customization of the plot.

`afex_plot` is the user friendly function that does data preparation and plotting. It also allows to only return the prepared data (`return = "data"`).

`interaction_plot` does the plotting when a `trace` factor is present. `oneway_plot` does the plotting when a `trace` factor is absent.

Usage

```
afex_plot(object, ...)

## S3 method for class 'afex_aov'
afex_plot(object, x, trace, panel, mapping,
  error = "model", error_ci = TRUE, error_level = 0.95,
  error_arg = list(width = 0), data_plot = TRUE, data_geom,
  data_alpha = 0.5, data_arg = list(color = "darkgrey"),
  point_arg = list(), line_arg = list(), emmeans_arg = list(),
  dodge = 0.5, return = "plot", factor_levels = list(), legend_title,
  ...)

## S3 method for class 'mixed'
afex_plot(object, x, trace, panel, mapping, id,
  error = "model", error_ci = TRUE, error_level = 0.95,
  error_arg = list(width = 0), data_plot = TRUE, data_geom,
  data_alpha = 0.5, data_arg = list(color = "darkgrey"),
  point_arg = list(), line_arg = list(), emmeans_arg = list(),
  dodge = 0.5, return = "plot", factor_levels = list(), legend_title,
  ...)

## S3 method for class 'merMod'
afex_plot(object, x, trace, panel, mapping, id,
  error = "model", error_ci = TRUE, error_level = 0.95,
  error_arg = list(width = 0), data_plot = TRUE, data_geom,
  data_alpha = 0.5, data_arg = list(color = "darkgrey"),
  point_arg = list(), line_arg = list(), emmeans_arg = list(),
  dodge = 0.5, return = "plot", factor_levels = list(), legend_title,
  ...)

## Default S3 method:
```

```
afex_plot(object, x, trace, panel, mapping, id, dv, data,
  within_vars, between_vars, error = "model", error_ci = TRUE,
  error_level = 0.95, error_arg = list(width = 0), data_plot = TRUE,
  data_geom, data_alpha = 0.5, data_arg = list(color = "darkgrey"),
  point_arg = list(), line_arg = list(), emmeans_arg = list(),
  dodge = 0.5, return = "plot", factor_levels = list(), legend_title,
  ...)

interaction_plot(means, data, mapping = c("shape", "lineytpe"),
  error_plot = TRUE, error_arg = list(width = 0), data_plot = TRUE,
  data_geom = ggplot2::geom_point, data_alpha = 0.5,
  data_arg = list(color = "darkgrey"), point_arg = list(),
  line_arg = list(), dodge = 0.5, legend_title, col_x = "x",
  col_y = "y", col_trace = "trace", col_panel = "panel",
  col_lower = "lower", col_upper = "upper")

oneway_plot(means, data, mapping = "", error_plot = TRUE,
  error_arg = list(width = 0), data_plot = TRUE,
  data_geom = ggbeeswarm::geom_beeswarm, data_alpha = 0.5,
  data_arg = list(color = "darkgrey"), point_arg = list(),
  legend_title, col_x = "x", col_y = "y", col_panel = "panel",
  col_lower = "lower", col_upper = "upper")
```

Arguments

object	afex_aov, mixed, merMod or other model object supported by **emmeans** (for further examples see: vignette("afex_plot_supported_models")).
...	currently ignored.
x	A character vector or one-sided formula specifying the factor names of the predictors displayed on the x-axis. mapping specifies further mappings for these factors if trace is missing.
trace	An optional character vector or one-sided formula specifying the factor names of the predictors connected by the same line. mapping specifies further mappings for these factors.
panel	An optional character vector or one-sided formula specifying the factor names of the predictors shown in different panels.
mapping	A character vector specifying which aesthetic mappings should be applied to either the trace factors (if trace is specified) or the x factors. Useful options are any combination of "shape", "color", "linetype", or also "fill" (see examples). The default (i.e., missing) uses c("shape","linetype") if trace is specified and "" otherwise (i.e., no additional aesthetic). If specific mappings should not be applied to specific graphical elements, one can override those via the corresponding further arguments. For example, for data_arg the default is list(color = "darkgrey") which prevents that "color" is mapped onto points in the background.
error	A scalar character vector specifying on which standard error the error bars should be based. Default is "model", which plots model-based standard er-

rors. Further options are: "none" (or NULL), "mean", "within" (or "CMO"), and "between". See details.

error_ci — Logical. Should error bars plot confidence intervals (=TRUE, the default) or standard errors (=FALSE)?

error_level — Numeric value between 0 and 1 determing the width of the confidence interval. Default is .95 corresponding to a 95% confidence interval.

error_arg — A list of further arguments passed to geom_errorbar, which draws the errorsbars. Default is list(width = 0) which suppresses the vertical bars at the end of the error bar.

data_plot — logical. Should raw data be plotted in the background? Default is TRUE.

data_geom — Geom function used for plotting data in background. The default (missing) uses geom_point if trace is specified, otherwise geom_beeswarm. See examples fo further options.

data_alpha — numeric alpha value between 0 and 1 passed to data_geom. Default is 0.5 which correspond to semitransparent data points in the background such that overlapping data points are plotted darker.

data_arg — A list of further arguments passed to data_geom. Default is list(color = "darkgrey"), which plots points in the background in grey.

point_arg, line_arg — A list of further arguments passed to geom_point or geom_line which draw the points and lines in the foreground. Default is list(). line_arg is only used if trace is specified.

emmeans_arg — A list of further arguments passed to emmeans. Of particular importance for ANOVAs is model, see afex_aov-methods.

dodge — Numerical amount of dodging of factor-levels on x-axis. Default is 0.5.

return — A scalar character specifying what should be returned. The default "plot" returns the **ggplot2** plot. The other option "data" returns a list with two data.frames containing the data used for plotting: means contains the means and standard errors for the foreground, data contains the raw data in the background.

factor_levels — A list of new factor levels that should be used in the plot. The name of each list entry needs to correspond to one of the factors in the plot.

legend_title — A scalar character vector with a new title for the legend.

id — An optional character vector specifying over which variables the raw data should be aggregated. Only relevant for mixed, merMod, and default method. The default (missing) uses all random effects grouping factors (for mixed and merMod method) or assumes all data points are independent. This can lead to many data points. error = "within" or error = "between" require that id is of length 1. See examples.

dv — An optional scalar character vector giving the name of the column containing the dependent variable for the afex_plot.default method. If missing, the function attempts to take it from the call slot of object. This is also used as y-axis label.

`data` For the `afex_plot.default` method, an optional `data.frame` containing the raw data used for fitting the model and which will be used as basis for the data points in the background. If missing, it will be attempted to obtain it from the model via `recover_data`. For the plotting functions, a `data.frame` with the data that has to be passed and contains the background data points.

`within_vars, between_vars`
For the `afex_plot.default` method, an optional `character` vector specifying which variables should be treated as within-subjects (or repeated-measures) factors and which as between-subjects (or independen-sampels) factors. If one of the two arguments is given, all other factors are assumed to fall into the other category.

`means` `data.frame`s used for plotting of the plotting functions.

`error_plot` `logical`. Should error bars be plotted? Only used in plotting functions. To suppress plotting of error bars use `error = "none"` in `afex_plot`.

`col_y, col_x, col_trace, col_panel`
A scalar `character` string specifying the name of the corresponding column containing the information used for plotting. Each column needs to exist in both the `means` and the `data` `data.frame`.

`col_lower, col_upper`
A scalar `character` string specifying the name of the columns containing lower and upper bounds for the error bars. These columns need to exist in `means`.

Details

`afex_plot` obtains the estimated marginal means via `emmeans` and aggregates the raw data to the same level. It then calculates the desired confidence interval or standard error (see below) and passes the prepared data to one of the two plotting functions: `interaction_plot` when `trace` is specified and `oneway_plot` otherwise.

Error Bars: Error bars provide a grahical representation of the variability of the estimated means and should be routinely added to results figures. However, there exist several possibilities which particular measure of variability to use. Because of this, any figure depicting error bars should be accompanied by a note detailing which measure the error bars shows. The present functions allow plotting of different types of confidence intervals (if `error_ci = TRUE`, the default) or standard errors (if `error_ci = FALSE`).

A further complication is that readers routinely misinterpret confidence intervals. The most common error is to assume that non-overlapping error bars indicate a significant difference (e.g., Belia et al., 2005). This is rarely the case (see e.g., Cumming & Finch, 2005; Knol et al., 2011; Schenker & Gentleman, 2005). For example, in a fully between-subjects design in which the error bars depict 95% confidence intervals and groups are of approximately equal size and have equal variance, even error bars that overlap by as much as 50% still correspond to $p < .05$. Error bars that are just touching roughly correspond to $p = .01$.

In the case of designs involving repeated-measures factors the usual confidence intervals or standard errors (i.e., model-based confidence intervals or intervals based on the standard error of the mean) cannot be used to gauge significant differences as this requires knowledge about the correlation between measures. One popular alternative in the psychological literature are intervals based on within-subjects standard errors/confidence intervals (e.g., Cousineau & O'Brien, 2014).

These attempt to control for the correlation across individuals and thereby allow judging differences between repeated-measures condition. As a downside, when using within-subjects intervals no comparisons across between-subjects conditions or with respect to a fixed-value are possible anymore.

In the case of a mixed-design, no single type of error bar is possible that allows comparison across all conditions. Likewise, for mixed models involving multiple *crossed* random effects, no single set of error bars (or even data aggregation) adequately represent the true variability in the data and adequately allows for "inference by eye". Therefore, special care is necessary in such cases. One possiblity is to avoid error bars altogether and plot only the raw data in the background (with `error = "none"`). The raw data in the background still provides a visual impression of the variability in the data and the precision of the mean estimate, but does not as easily suggest an incorrect inferences. Another possibility is to use the model-based standard error and note in the figure caption that it does not permit comparisons across repeated-measures factors.

The following "rules of eye" (Cumming and Finch, 2005) hold, when permitted by design (i.e., within-subjects bars for within-subjects comparisons; other variants for between-subjects comparisons), and groups are approximately equal in size and variance. Note that for more complex designs ususally analyzed with mixed models, such as designs involving complicated dependencies across data points, these rules of thumbs may be highly misleading.

- $p < .05$ when the overlap of the 95% confidence intervals (CIs) is no more than about half the average margin of error, that is, when proportion overlap is about .50 or less.
- $p < .01$ when the two CIs do not overlap, that is, when proportion overlap is about 0 or there is a positive gap.
- $p < .05$ when the gap between standard error (SE) bars is at least about the size of the average SE, that is, when the proportion gap is about 1 or greater.
- $p < .01$ when the proportion gap between SE bars is about 2 or more.

Implemented Standard Errors: The following lists the implemented approaches to calculate confidence intervals (CIs) and standard errors (SEs). CIs are based on the SEs using the *t*-distribution with degrees of freedom based on the cell or group size. For ANOVA models, `afex_plot` attempts to warn in case the chosen approach is misleading given the design (e.g., model-based error bars for purely within-subjects plots). For `mixed` models, no such warnings are produced, but users should be aware that all options beside `"model"` are not actually appropriate and have only heuristic value. But then again, `"model"` based error bars do not permit comparisons for factors varying within one of the random-effects grouping factors (i.e., factors for which random-slopes should be estimated).

`"model"` Uses model-based CIs and SEs. For ANOVAs, the variant based on the `lm` or `mlm` model (i.e., `emmeans_arg = list(model = "multivariate")`) seems generally preferrable.

`"mean"` Calculates the standard error of the mean for each cell ignoring any repeated-measures factors.

`"within"` **or** `"CMO"` Calculates within-subjects SEs using the Cosineau-Morey-O'Brien (Cousineau & O'Brien, 2014) method. This method is based on a double normalization of the data. SEs and CIs are then calculated independently for each cell (i.e., if the desired output contains between-subjects factors, SEs are calculated for each cell including the between-subjects factors).

`"between"` First aggregates the data per participant and then calculates the SEs for each between-subjects condition. Results in one SE and *t*-quantile for all conditions in purely within-subjects designs.

`"none"` **or** `NULL` Suppresses calculation of SEs and plots no error bars.

For `mixed` models, the within-subjects/repeated-measures factors are relative to the chosen `id` effects grouping factor. They are automatically detected based on the random-slopes of the random-effects grouping factor in `id`. All other factors are treated as independent-samples or between-subjects factors.

Value

Returns a **ggplot2** plot (i.e., object of class `c("gg","ggplot")`) unless `return = "data"`.

References

Belia, S., Fidler, F., Williams, J., & Cumming, G. (2005). Researchers Misunderstand Confidence Intervals and Standard Error Bars. *Psychological Methods*, 10(4), 389-396. https://doi.org/10.1037/1082-989X.10.4.389

Cousineau, D., & O'Brien, F. (2014). Error bars in within-subject designs: a comment on Baguley (2012). *Behavior Research Methods*, 46(4), 1149-1151. https://doi.org/10.3758/s13428-013-0441-z

Cumming, G., & Finch, S. (2005). Inference by Eye: Confidence Intervals and How to Read Pictures of Data. *American Psychologist*, 60(2), 170-180. https://doi.org/10.1037/0003-066X.60.2.170

Knol, M. J., Pestman, W. R., & Grobbee, D. E. (2011). The (mis)use of overlap of confidence intervals to assess effect modification. *European Journal of Epidemiology*, 26(4), 253-254. https://doi.org/10.1007/s10654-011-9563-8

Schenker, N., & Gentleman, J. F. (2001). On Judging the Significance of Differences by Examining the Overlap Between Confidence Intervals. *The American Statistician*, 55(3), 182-186. https://doi.org/10.1198/000313001317097960

Examples

```
# note: use library("ggplot") to avoid "ggplot2::" in the following

######################################################################
##                2-factor Within-Subject Design                    ##
######################################################################

data(md_12.1)
aw <- aov_ez("id", "rt", md_12.1, within = c("angle", "noise"))

##----------------------------------------------------------------
##                    Basic Interaction Plots                     -
##----------------------------------------------------------------

afex_plot(aw, x = "angle", trace = "noise")
# or: afex_plot(aw, x = ~angle, trace = ~noise)

afex_plot(aw, x = "noise", trace = "angle")

### For within-subject designs, using within-subject CIs is better:
afex_plot(aw, x = "angle", trace = "noise", error = "within")
```

```
(p1 <- afex_plot(aw, x = "noise", trace = "angle", error = "within"))

## use different themes for nicer graphs:
p1 + ggplot2::theme_bw()
## Not run:
p1 + ggplot2::theme_light()
p1 + ggplot2::theme_minimal()
p1 + jtools::theme_apa()
p1 + ggpubr::theme_pubr()

### set theme globally for R session:
ggplot2::theme_set(ggplot2::theme_bw())

### There are several ways to deal with overlapping points in the background besides alpha
# 1. using the default data geom and ggplot2::position_jitterdodge
afex_plot(aw, x = "noise", trace = "angle", error = "within", dodge = 0.3,
          data_arg = list(
            position =
              ggplot2::position_jitterdodge(
                jitter.width = 0,
                jitter.height = 5,
                dodge.width = 0.3  ## needs to be same as dodge
                ),
            color = "darkgrey"))

# 2. using ggbeeswarm::geom_beeswarm
afex_plot(aw, x = "noise", trace = "angle", error = "within", dodge = 0.5,
          data_geom = ggbeeswarm::geom_beeswarm,
          data_arg = list(
            dodge.width = 0.5,  ## needs to be same as dodge
            cex = 0.8,
            color = "darkgrey"))

# 3. do not display points, but use a violinplot: ggplot2::geom_violin
afex_plot(aw, x = "noise", trace = "angle", error = "within",
          data_geom = ggplot2::geom_violin,
          data_arg = list(width = 0.5))

# 4. violinplots with color: ggplot2::geom_violin
afex_plot(aw, x = "noise", trace = "angle", error = "within",
          mapping = c("linetype", "shape", "fill"),
          data_geom = ggplot2::geom_violin,
          data_arg = list(width = 0.5))

# 5. do not display points, but use a boxplot: ggplot2::geom_boxplot
afex_plot(aw, x = "noise", trace = "angle", error = "within",
          data_geom = ggplot2::geom_boxplot,
          data_arg = list(width = 0.3))

# 6. combine points with boxplot: ggpol::geom_boxjitter
afex_plot(aw, x = "noise", trace = "angle", error = "within",
          data_geom = ggpol::geom_boxjitter,
          data_arg = list(width = 0.3))
```

```
## hides error bars!

# 7. nicer variant of ggpol::geom_boxjitter
afex_plot(aw, x = "noise", trace = "angle", error = "within",
          mapping = c("shape", "fill"),
          data_geom = ggpol::geom_boxjitter,
          data_arg = list(
            width = 0.3,
            jitter.width = 0,
            jitter.height = 10,
            outlier.intersect = TRUE),
          point_arg = list(size = 2.5),
          error_arg = list(size = 1.5, width = 0))

# 8. nicer variant of ggpol::geom_boxjitter without lines
afex_plot(aw, x = "noise", trace = "angle", error = "within", dodge = 0.7,
          mapping = c("shape", "fill"),
          data_geom = ggpol::geom_boxjitter,
          data_arg = list(
            width = 0.5,
            jitter.width = 0,
            jitter.height = 10,
            outlier.intersect = TRUE),
          point_arg = list(size = 2.5),
          line_arg = list(linetype = 0),
          error_arg = list(size = 1.5, width = 0))

## End(Not run)


##------------------------------------------------------------------
##                        One-Way Plots                           -
##------------------------------------------------------------------

afex_plot(aw, x = "angle", error = "within") ## default

## Not run:
## with color we need larger points
afex_plot(aw, x = "angle", mapping = "color", error = "within",
          point_arg = list(size = 2.5),
          error_arg = list(size = 1.5, width = 0.05))

afex_plot(aw, x = "angle", error = "within", data_geom = ggpol::geom_boxjitter)

## nicer
afex_plot(aw, x = "angle", error = "within", data_geom = ggpol::geom_boxjitter,
          mapping = "fill", data_alpha = 0.7,
          data_arg = list(
            width = 0.6,
            jitter.width = 0.07,
            jitter.height = 10,
            outlier.intersect = TRUE
          ),
```

```
          point_arg = list(size = 2.5),
          error_arg = list(size = 1.5, width = 0.05))

## we can add a line connecting the means using geom_point(aes(group = 1)):
afex_plot(aw, x = "angle", error = "within") +
  ggplot2::geom_line(ggplot2::aes(group = 1))

## One-way plots also supports panels:
afex_plot(aw, x = "angle", panel = "noise", error = "within")

## And panels with lines:
afex_plot(aw, x = "angle", panel = "noise", error = "within") +
  ggplot2::geom_line(ggplot2::aes(group = 1))


## For more complicated plots it is easier to attach ggplot2:
library("ggplot2")

## We can hide geoms by plotting them in transparent color and add them
## afterward to use a mapping not directly supported.
## For example, the next plot adds a line to a one-way plot with panels, but
## with all geoms in the foreground having a color conditional on the panel.

afex_plot(aw, x = "angle", panel = "noise", error = "within",
          point_arg = list(color = "transparent"),
          error_arg = list(color = "transparent")) +
  geom_point(aes(color = panel)) +
  geom_linerange(aes(color = panel, ymin = lower, ymax = upper)) +
  geom_line(aes(group = 1, color = panel)) +
  guides(color = guide_legend(title = "NOISE"))
## Note that we need to use guides explicitly, otherwise the legend title would
## be "panel". legend_title does not work in this case.

##------------------------------------------------------------
##                     Other Basic Options                    -
##------------------------------------------------------------

## relabel factor levels via factor_levels
afex_plot(aw, x = "noise", trace = "angle",
          factor_levels = list(angle = c("0°", "4°", "8°"),
                               noise = c("Absent", "Present")))

## Change title of legend
afex_plot(aw, x = "noise", trace = "angle",
          legend_title = "Noise Condition")

## for plots with few factor levels, smaller dodge might be better:
afex_plot(aw, x = "angle", trace = "noise", dodge = 0.25)

######################################################################
##                   4-factor Mixed Design                    ##
######################################################################
```

```
data(obk.long, package = "afex")
a1 <- aov_car(value ~ treatment * gender + Error(id/(phase*hour)),
              data = obk.long, observed = "gender")

## too difficult to see anything
afex_plot(a1, ~phase*hour, ~treatment) +
  ggplot2::theme_light()

## better
afex_plot(a1, ~hour, ~treatment, ~phase) +
  ggplot2::theme_light()

## even better and different model-based standard errors
afex_plot(a1, ~hour, ~treatment, ~phase,
          dodge = 0.65,
          data_arg = list(
            position =
              ggplot2::position_jitterdodge(
                jitter.width = 0,
                jitter.height = 0.2,
                dodge.width = 0.65  ## needs to be same as dodge
                ),
            color = "darkgrey"),
          emmeans_arg = list(model = "multivariate")) +
  ggplot2::theme_classic()

# with color instead of linetype to separate trace factor
afex_plot(a1, ~hour, ~treatment, ~phase,
          mapping = c("shape", "color"),
          dodge = 0.65,
          data_arg = list(
            position =
              ggplot2::position_jitterdodge(
                jitter.width = 0,
                jitter.height = 0.2,
                dodge.width = 0.65  ## needs to be same as dodge
                )),
          emmeans_arg = list(model = "multivariate")) +
  ggplot2::theme_light()

# only color to separate trace factor
afex_plot(a1, ~hour, ~treatment, ~phase,
          mapping = "color",
          dodge = 0.65,
          data_arg = list(
            position =
              ggplot2::position_jitterdodge(
                jitter.width = 0,
                jitter.height = 0.2,
                dodge.width = 0.65  ## needs to be same as dodge
                )),
          emmeans_arg = list(model = "multivariate")) +
  ggplot2::theme_classic()
```

```
## plot involving all 4 factors:
afex_plot(a1, ~hour, ~treatment, ~gender+phase,
          dodge = 0.65,
          data_arg = list(
            position =
              ggplot2::position_jitterdodge(
                jitter.width = 0,
                jitter.height = 0.2,
                dodge.width = 0.65  ## needs to be same as dodge
                ),
            color = "darkgrey"),
          emmeans_arg = list(model = "multivariate")) +
  ggplot2::theme_bw()


##------------------------------------------------------------
##              Different Standard Errors Available           -
##------------------------------------------------------------

## purely within-design
cbind(
  afex_plot(a1, ~phase, ~hour,
            error = "model", return = "data")$means[,c("phase", "hour", "y", "SE")],
  multivariate = afex_plot(a1, ~phase, ~hour,
                           emmeans_arg = list(model = "multivariate"),
                           error = "model", return = "data")$means$error,
  mean = afex_plot(a1, ~phase, ~hour,
                   error = "mean", return = "data")$means$error,
  within = afex_plot(a1, ~phase, ~hour,
                     error = "within", return = "data")$means$error,
  between = afex_plot(a1, ~phase, ~hour,
                      error = "between", return = "data")$means$error)
## mixed design
cbind(
  afex_plot(a1, ~phase, ~treatment,
            error = "model", return = "data")$means[,c("phase", "treatment", "y", "SE")],
  multivariate = afex_plot(a1, ~phase, ~treatment,
                           emmeans_arg = list(model = "multivariate"),
                           error = "model", return = "data")$means$error,
  mean = afex_plot(a1, ~phase, ~treatment,
                   error = "mean", return = "data")$means$error,
  within = afex_plot(a1, ~phase, ~treatment,
                     error = "within", return = "data")$means$error,
  between = afex_plot(a1, ~phase, ~treatment,
                      error = "between", return = "data")$means$error)

## End(Not run)

######################################################################
##                           Mixed Models                           ##
######################################################################
```

```
data("Machines", package = "MEMSS")
m1 <- mixed(score ~ Machine + (Machine|Worker), data=Machines)

pairs(emmeans::emmeans(m1, "Machine"))
# contrast   estimate       SE df t.ratio p.value
# A - B     -7.966667 2.420850  5  -3.291  0.0481
# A - C    -13.916667 1.540100  5  -9.036  0.0007
# B - C     -5.950000 2.446475  5  -2.432  0.1253

## Default (i.e., model-based) error bars suggest no difference between Machines.
## This contrasts with pairwise comparisons above.
afex_plot(m1, "Machine")

## Impression from within-subject error bars is more in line with pattern of differences.
afex_plot(m1, "Machine", error = "within")


## Not run:
data("fhch2010") # load
fhch <- droplevels(fhch2010[ fhch2010$correct,]) # remove errors
### following model should take less than a minute to fit:
mrt <- mixed(log_rt ~ task*stimulus*frequency + (stimulus*frequency||id)+
               (task||item), fhch, method = "S", expand_re = TRUE)

## way too many points in background:
afex_plot(mrt, "stimulus", "frequency", "task")

## better to restrict plot of data to one random-effects grouping variable
afex_plot(mrt, "stimulus", "frequency", "task", id = "id")
## when plotting data from a single random effect, different error bars are possible:
afex_plot(mrt, "stimulus", "frequency", "task", id = "id", error = "within")
afex_plot(mrt, "stimulus", "frequency", "task", id = "id", error = "mean")

## compare visual impression with:
pairs(emmeans::emmeans(mrt, c("stimulus", "frequency"), by = "task"))

## same logic also possible for other random-effects grouping factor
afex_plot(mrt, "stimulus", "frequency", "task", id = "item")
## within-item error bars are misleading here. task is sole within-items factor.
afex_plot(mrt, "stimulus", "frequency", "task", id = "item", error = "within")
## CIs based on stanard error of mean look small, but not unreasonable given results.
afex_plot(mrt, "stimulus", "frequency", "task", id = "item", error = "mean")

### compare distribution of individual data for different random effects:
## requires package cowplot
p_id <- afex_plot(mrt, "stimulus", "frequency", "task", id = "id",
                  error = "within", dodge = 0.7,
                  data_geom = ggplot2::geom_violin,
                  mapping = c("shape", "fill"),
                  data_arg = list(width = 0.7)) +
  ggplot2::scale_shape_manual(values = c(4, 17)) +
  ggplot2::labs(title = "ID")
```

```
p_item <- afex_plot(mrt, "stimulus", "frequency", "task", id = "item",
          error = "within", dodge = 0.7,
          data_geom = ggplot2::geom_violin,
          mapping = c("shape", "fill"),
          data_arg = list(width = 0.7)) +
  ggplot2::scale_shape_manual(values = c(4, 17)) +
  ggplot2::labs(title = "Item")

### see: https://cran.r-project.org/package=cowplot/vignettes/shared_legends.html
p_comb <- cowplot::plot_grid(
  p_id + ggplot2::theme_light() + ggplot2::theme(legend.position="none"),
  p_item + ggplot2::theme_light() + ggplot2::theme(legend.position="none")
  )
legend <- cowplot::get_legend(p_id + ggplot2::theme(legend.position="bottom"))
cowplot::plot_grid(p_comb, legend,
                   ncol = 1,
                   rel_heights = c(1, 0.1))

##------------------------------------------------------------------
##                    Support for lme4::lmer                      -
##------------------------------------------------------------------

Oats <- nlme::Oats
## afex_plot does currently not support implicit nesting: (1|Block/Variety)
## Instead, we need to create the factor explicitly
Oats$VarBlock <- Oats$Variety:Oats$Block
Oats.lmer <- lmer(yield ~ Variety * factor(nitro) + (1|VarBlock) + (1|Block),
                        data = Oats)
afex_plot(Oats.lmer, "nitro", "Variety")
afex_plot(Oats.lmer, "nitro", panel = "Variety")

######################################################################
##     Default Method works for Models Supported by emmeans     ##
######################################################################

## lm
warp.lm <- lm(breaks ~ wool * tension, data = warpbreaks)
afex_plot(warp.lm, "tension")
afex_plot(warp.lm, "tension", "wool")

## poisson glm
ins <- data.frame(
    n = c(500, 1200, 100, 400, 500, 300),
    size = factor(rep(1:3,2), labels = c("S","M","L")),
    age = factor(rep(1:2, each = 3)),
    claims = c(42, 37, 1, 101, 73, 14))
ins.glm <- glm(claims ~ size + age + offset(log(n)),
               data = ins, family = "poisson")
afex_plot(ins.glm, "size", "age")

## binomial glm adapted from ?predict.glm
ldose <- factor(rep(0:5, 2))
```

```
numdead <- c(1, 4, 9, 13, 18, 20, 0, 2, 6, 10, 12, 16)
sex <- factor(rep(c("M", "F"), c(6, 6)))
SF <- numdead/20  ## dv should be a vector, no matrix
budworm.lg <- glm(SF ~ sex*ldose, family = binomial,
                  weights = rep(20, length(numdead)))
afex_plot(budworm.lg, "ldose")
afex_plot(budworm.lg, "ldose", "sex") ## data point is hidden behind mean!
afex_plot(budworm.lg, "ldose", "sex",
          data_arg = list(size = 4, color = "red"))

## nlme mixed model
data(Oats, package = "nlme")
Oats$nitro <- factor(Oats$nitro)
oats.1 <- nlme::lme(yield ~ nitro * Variety,
                    random = ~ 1 | Block / Variety,
                    data = Oats)
afex_plot(oats.1, "nitro", "Variety", data = Oats)
afex_plot(oats.1, "nitro", "Variety", data = Oats, id = "Block")
afex_plot(oats.1, "nitro", data = Oats)
afex_plot(oats.1, "nitro", data = Oats, id = c("Block", "Variety"))
afex_plot(oats.1, "nitro", data = Oats, id = "Block")


## End(Not run)
```

`all_fit` *Refit* `lmer` *model using multiple optimizers*

Description

Attempt to re-fit a [g]lmer model with a range of optimizers. The default is to use all known optimizers for R that satisfy the requirements (do not require explicit gradients, allow box constraints), in four categories; (i) built-in (`minqa::bobyqa`, `lme4::Nelder_Mead`), (ii) wrapped via optimx (most of optimx's optimizers that allow box constraints require an explicit gradient function to be specified; the two provided here are really base R functions that can be accessed via optimx, (iii) wrapped via nloptr, (iv) `dfoptim::nmkb`.

Usage

```
all_fit(m, meth_tab = cbind(optimizer = rep(c("bobyqa", "Nelder_Mead",
  "optimx", "nloptwrap", "nmkbw"), c(1, 1, 2, 2, 1)), method = c("", "",
  "nlminb", "L-BFGS-B", "NLOPT_LN_NELDERMEAD", "NLOPT_LN_BOBYQA", "")),
  verbose = TRUE, maxfun = 1e+06, ...)

nmkbw(fn, par, lower, upper, control)
```

Arguments

`m`	a fitted model with `lmer`

meth_tab	a matrix (or data.frame) with columns - method the name of a specific optimization method to pass to the optimizer (leave blank for built-in optimizers) - optimizer the optimizer function to use
verbose	print progress messages?
maxfun	number of iterations to allow for the optimization rountine.
...	further arguments passed to update.merMod such as data.
fn	needed for dfoptim::nmkb
par	needed for dfoptim::nmkb
lower	needed for dfoptim::nmkb
upper	needed for dfoptim::nmkb
control	needed for dfoptim::nmkb

Details

Needs packages **nloptr**, **optimx**, and dfoptim to try out all optimizers. **optimx** needs to be loaded explicitly using library or require (see examples).

nmkbw is a simple wrapper function for fitting models with the corresponding optimizer. It needs to be exported for lme4, but should not be called directly by the user.

Value

a list of fitted merMod objects

Note

Very similar to the function of the same name that is part of **lme4**. The present function will be removed eventually in favor of the **lme4** function.

Author(s)

Ben Bolker, minor changes by Henrik Singmann

See Also

slice, slice2D in the bbmle package

Examples

```
## Not run:

# basic usage
require(optimx)
gm1 <- glmer(cbind(incidence, size - incidence) ~ period + (1 | herd),
             data = cbpp, family = binomial)
gm_all <- all_fit(gm1)
t(sapply(gm_all,fixef)) ## extract fixed effects
sapply(gm_all,logLik) ## log-likelihoods
```

```
sapply(gm_all,getME,"theta") ## theta parameters
!sapply(gm_all,inherits,"try-error") ## was fit OK?


## for GLMMs:
require("mlmRev") # for data
gm1 <- mixed(use ~ age*urban + (1 | district), family = binomial,
             data = Contraception, method = "LRT")
gm_all <- all_fit(gm1$full_model)
sapply(gm_all,logLik)

## use allFit in combination with expand.re = TRUE
data("sk2011.2") # see example("mixed")
sk2_aff <- droplevels(sk2011.2[sk2011.2$what == "affirmation",])
sk_m2 <- mixed(response ~ instruction*inference*type+(inference*type||id), sk2_aff,
               expand_re = TRUE)
sk_m2
sk_m2_allFit <- all_fit(sk_m2$full_model)
sk_m2_allFit # all fits fail

sk_m2_allFit <- all_fit(sk_m2$full_model, data = sk_m2$data) # works
t(sapply(sk_m2_allFit,fixef))
sapply(sk_m2_allFit,logLik)


## End(Not run)
```

---

aov_car *Convenient ANOVA estimation for factorial designs*

Description

These functions allow convenient specification of any type of ANOVAs (i.e., purely within-subjects ANOVAs, purely between-subjects ANOVAs, and mixed between-within or split-plot ANOVAs) for data in the **long** format (i.e., one observation per row). If the data has more than one observation per individual and cell of the design (e.g., multiple responses per condition), the data will by automatically aggregated. The default settings reproduce results from commercial statistical packages such as SPSS or SAS. `aov_ez` is called specifying the factors as character vectors, `aov_car` is called using a formula similar to `aov` specifying an error strata for the within-subject factor(s), and `aov_4` is called with a **lme4**-like formula (all ANOVA functions return identical results). The returned object contains the ANOVA also fitted via base R's `aov` which can be passed to e.g., **emmeans** for further analysis (e.g., follow-up tests, contrasts, plotting, etc.). These functions employ `Anova` (from the **car** package) to provide test of effects avoiding the somewhat unhandy format of `car::Anova`.

Usage

```
aov_car(formula, data, fun_aggregate = NULL,
  type = afex_options("type"), factorize = afex_options("factorize"),
  check_contrasts = afex_options("check_contrasts"), observed = NULL,
```

```
  anova_table = list(), include_aov = afex_options("include_aov"),
  return = afex_options("return_aov"), ...)

aov_4(formula, data, observed = NULL, fun_aggregate = NULL,
  type = afex_options("type"), factorize = afex_options("factorize"),
  check_contrasts = afex_options("check_contrasts"),
  return = afex_options("return_aov"), anova_table = list(),
  include_aov = afex_options("include_aov"), ...,
  print.formula = FALSE)

aov_ez(id, dv, data, between = NULL, within = NULL, covariate = NULL,
  observed = NULL, fun_aggregate = NULL, transformation,
  type = afex_options("type"), factorize = afex_options("factorize"),
  check_contrasts = afex_options("check_contrasts"),
  return = afex_options("return_aov"), anova_table = list(),
  include_aov = afex_options("include_aov"), ...,
  print.formula = FALSE)
```

Arguments

formula	A formula specifying the ANOVA model similar to aov (for aov_car or similar to lme4:lmer for aov_4). Should include an error term (i.e., Error(id/...) for aov_car or (...|id) for aov_4). Note that the within-subject factors do not need to be outside the Error term (this contrasts with aov). See Details.
data	A data.frame containing the data. Mandatory.
fun_aggregate	The function for aggregating the data before running the ANOVA if there is more than one observation per individual and cell of the design. The default NULL issues a warning if aggregation is necessary and uses mean. Pass mean directly to avoid the warning.
type	The type of sums of squares for the ANOVA. The default is given by afex_options("type"), which is **initially set to 3**. Passed to Anova. Possible values are "II", "III", 2, or 3.
factorize	logical. Should between subject factors be factorized (with note) before running the analysis. The default is given by afex_options("factorize"), which is initially TRUE. If one wants to run an ANCOVA, this needs to be set to FALSE (in which case centering on 0 is checked on numeric variables).
check_contrasts	logical. Should contrasts for between-subject factors be checked and (if necessary) changed to be "contr.sum". See details. The default is given by afex_options("check_contrasts"), which is initially TRUE.
observed	character vector indicating which of the variables are observed (i.e, measured) as compared to experimentally manipulated. The default effect size reported (generalized eta-squared) requires correct specification of the obsered (in contrast to manipulated) variables.
anova_table	list of further arguments passed to function producing the ANOVA table. Arguments such as es (effect size) or correction are passed to either anova.afex_aov

	or `nice`. Note that those settings can also be changed once an object of class `afex_aov` is created by invoking the `anova` method directly.
`include_aov`	Boolean. Allows suppressing the calculation of the aov object, which is per default part of the returned `afex_aov` object. `FALSE` prevents this potentially costly calculation. Especially for designs with larger N and within-subjects factors, this is highly advisable. Follow-up analyses using **emmeans** are then always based on the multivariate or `lm` model.
`return`	What should be returned? The default is given by `afex_options("return_aov")`, which is initially `"afex_aov"`, returning an S3 object of class `afex_aov` for which various methods exist (see there and below for more details). Other values are currently still supported for backward compatibility.
`...`	Further arguments passed to `fun_aggregate`.
`print.formula`	`aov_ez` and `aov_4` are wrapper for `aov_car`. This boolean argument indicates whether the formula in the call to `car.aov` should be printed.
`id`	`character` vector (of length 1) indicating the subject identifier column in `data`.
`dv`	`character` vector (of length 1) indicating the column containing the **dependent variable** in `data`.
`between`	`character` vector indicating the **between**-subject(s) factor(s)/column(s) in `data`. Default is `NULL` indicating no between-subjects factors.
`within`	`character` vector indicating the **within**-subject(s)(or repeated-measures) factor(s)/column(s) in `data`. Default is `NULL` indicating no within-subjects factors.
`covariate`	`character` vector indicating the between-subject(s) covariate(s) (i.e., column(s)) in `data`. Default is `NULL` indicating no covariates. Please note that `factorize` needs to be set to `FALSE` in case the covariate is numeric and should be treated as such.
`transformation`	In `aov_ez`, a `character` vector (of length 1) indicating the name of a transformation to apply to `dv` before fitting the model. If missing, no transformation is applied. In `aov_car` and `aov_4`, a response transformation may be incorporated in the left-hand side of `formula`.

Details

Details of ANOVA Specification: `aov_ez` will concatenate all between-subject factors using `*` (i.e., producing all main effects and interactions) and all covariates by `+` (i.e., adding only the main effects to the existing between-subject factors). The within-subject factors do fully interact with all between-subject factors and covariates. This is essentially identical to the behavior of SPSS's `glm` function.

The `formula`s for `aov_car` or `aov_4` must contain a single `Error` term specifying the `ID` column and potential within-subject factors (you can use `mixed` for running mixed-effects models with multiple error terms). Factors outside the `Error` term are treated as between-subject factors (the within-subject factors specified in the `Error` term are ignored outside the `Error` term; in other words, it is not necessary to specify them outside the `Error` term, see Examples).
Suppressing the intercept (i.e, via `0 +` or `-1`) is ignored. Specific specifications of effects (e.g., excluding terms with `-` or using `^`) could be okay but is not tested. Using the `I` or `poly` function within the formula is not tested and not supported!

To run an ANCOVA you need to set `factorize = FALSE` and make sure that all variables have the correct type (i.e., factors are factors and numeric variables are numeric and centered).

Note that the default behavior is to include calculation of the effect size generalized eta-squared for which **all non-manipluated (i.e., observed)** variables need to be specified via the `observed` argument to obtain correct results. When changing the effect size to `"pes"` (partial eta-squared) or `"none"` via `anova_table` this becomes unnecessary.

If `check_contrasts = TRUE`, contrasts will be set to `"contr.sum"` for all between-subject factors if default contrasts are not equal to `"contr.sum"` or `attrib(factor,"contrasts") != "contr.sum"`. (within-subject factors are hard-coded `"contr.sum"`.)

Statistical Issues: Type 3 sums of squares are default in afex. While some authors argue that so-called type 3 sums of squares are dangerous and/or problematic (most notably Venables, 2000), they are the default in many commercial statistical application such as SPSS or SAS. Furthermore, statisticians with an applied perspective recommend type 3 tests (e.g., Maxwell and Delaney, 2004). Consequently, they are the default for the ANOVA functions described here. For some more discussion on this issue see here.

Note that lower order effects (e.g., main effects) in type 3 ANOVAs are only meaningful with effects coding. That is, contrasts should be set to `contr.sum` to obtain meaningful results. This is imposed automatically for the functions discussed here as long as `check_contrasts` is `TRUE` (the default). I nevertheless recommend to set the contrasts globally to `contr.sum` via running `set_sum_contrasts`. For a discussion of the other (non-recommended) coding schemes see here.

Follow-Up Contrasts and Post-Hoc Tests: The S3 object returned per default can be directly passed to `emmeans::emmeans` for further analysis. This allows to test any type of contrasts that might be of interest independent of whether or not this contrast involves between-subject variables, within-subject variables, or a combination thereof. The general procedure to run those contrasts is the following (see Examples for a full example):

1. Estimate an `afex_aov` object with the function returned here. For example: `x <-aov_car(dv ~ a*b + (id/c),d)`
2. Obtain a `emmGrid-class` object by running `emmeans` on the `afex_aov` object from step 1 using the factors involved in the contrast. For example: `r <-emmeans(x,~a:c)`
3. Create a list containing the desired contrasts on the reference grid object from step 2. For example: `con1 <-list(a_x = c(-1,1,0,0,0,0),b_x = c(0,0,-0.5,-0.5,0,1))`
4. Test the contrast on the reference grid using `contrast`. For example: `contrast(r,con1)`
5. To control for multiple testing p-value adjustments can be specified. For example the Bonferroni-Holm correction: `contrast(r,con1,adjust = "holm")`

Note that **emmeans** allows for a variety of advanced settings and simplifiations, for example: all pairwise comparison of a single factor using one command (e.g., `emmeans(x,"a",contr = "pairwise")`) or advanced control for multiple testing by passing objects to **multcomp**. A comprehensive overview of the functionality is provided in the accompanying vignettes (see here).

A caveat regarding the use of **emmeans** concerns the assumption of sphericity for ANOVAs including within-subjects/repeated-measures factors (with more than two levels). The current default for follow-up tests uses a univariate model (`model = "univariate"` in the call to `emmeans`), which does not adequately control for violations of sphericity. This may result in anti-conservative tests and contrasts somewhat with the default ANOVA table which reports results based on the Greenhousse-Geisser correction. An alternative is to use a multivariate model (`model = "multivariate"`

in the call to emmeans) which should handle violations of sphericity better. The default will likely change to multivariate tests in one of the next versions of the package.

Starting with **afex** version 0.22, **emmeans** is *not* loaded/attached automatically when loading **afex**. Therefore, **emmeans** now needs to be loaded by the user via `library("emmeans")` or `require("emmeans")`.

Methods for `afex_aov` **Objects:** A full overview over the methods provided for `afex_aov` objects is provided in the corresponding help page: `afex_aov-methods`. The probably most important ones for end-users are `summary`, `anova`, and `nice`.

The `summary` method returns, for ANOVAs containing within-subject (repeated-measures) factors with more than two levels, the complete univariate analysis: Results without df-correction, the Greenhouse-Geisser corrected results, the Hyunh-Feldt corrected results, and the results of the Mauchly test for sphericity.

The `anova` method returns a `data.frame` of class `"anova"` containing the ANOVA table in numeric form (i.e., the one in slot `anova_table` of a `afex_aov`). This method has arguments such as `correction` and `es` and can be used to obtain an ANOVA table with different correction than the one initially specified.

The `nice` method also returns a `data.frame`, but rounds most values and transforms them into characters for nice printing. Also has arguments like `correction` and `es` which can be used to obtain an ANOVA table with different correction than the one initially specified.

Value

`aov_car`, `aov_4`, and `aov_ez` are wrappers for `Anova` and `aov`, the return value is dependent on the `return` argument. Per default, an S3 object of class `"afex_aov"` is returned containing the following slots:

`"anova_table"` An ANOVA table of class `c("anova","data.frame")`.

`"aov"` `aov` object returned from `aov` (should not be used to evaluate significance of effects, but can be passed to `emmeans` for post-hoc tests).

`"Anova"` object returned from `Anova`, an object of class `"Anova.mlm"` (if within-subjects factors are present) or of class `c("anova","data.frame")`.

`"lm"` the object fitted with `lm` and passed to `Anova` (i.e., an object of class `"lm"` or `"mlm"`). Also returned if `return = "lm"`.

`"data"` a list containing: (1) `long` (the possibly aggregated data in long format used for `aov`), `wide` (the data used to fit the `lm` object), and `idata` (if within-subject factors are present, the `idata` argument passed to `car::Anova`). Also returned if `return = "data"`.

In addition, the object has the following attributes: `"dv"`, `"id"`, `"within"`, `"between"`, and `"type"`.

The print method for `afex_aov` objects (invisibly) returns (and prints) the same as if `return` is `"nice"`: a nice ANOVA table (produced by `nice`) with the following columns: `Effect`, `df`, `MSE` (mean-squared errors), `F` (potentially with significant symbols), `ges` (generalized eta-squared), `p`.

Functions

- `aov_4`: Allows definition of ANOVA-model using `lme4::lmer`-like Syntax (but still fits a standard ANOVA).
- `aov_ez`: Allows definition of ANOVA-model using character strings.

Note

Calculation of ANOVA models via aov (which is done per default) can be comparatively slow and produce comparatively large objects for ANOVAs with many within-subjects factors or levels. To avoid this calculation set include_aov = FALSE. You can also disable this globally with: afex_options(include_aov = FALSE)

The id variable and variables entered as within-subjects (i.e., repeated-measures) factors are silently converted to factors. Levels of within-subject factors are converted to valid variable names using make.names(...,unique=TRUE). Unused factor levels are silently dropped on all variables.

Contrasts attached to a factor as an attribute are probably not preserved and not supported.

The workhorse is aov_car. aov_4 and aov_ez only construe and pass an appropriate formula to aov_car. Use print.formula = TRUE to view this formula.

In contrast to aov aov_car assumes that all factors to the right of / in the Error term are belonging together. Consequently, Error(id/(a*b)) and Error(id/a*b) are identical (which is not true for aov).

Author(s)

Henrik Singmann

The design of these functions was influenced by ezANOVA from package **ez**.

References

Cramer, A. O. J., van Ravenzwaaij, D., Matzke, D., Steingroever, H., Wetzels, R., Grasman, R. P. P. P., ... Wagenmakers, E.-J. (2015). Hidden multiplicity in exploratory multiway ANOVA: Prevalence and remedies. *Psychonomic Bulletin & Review*, 1-8. doi:10.3758/s13423-015-0913-5

Maxwell, S. E., & Delaney, H. D. (2004). *Designing Experiments and Analyzing Data: A Model-Comparisons Perspective*. Mahwah, N.J.: Lawrence Erlbaum Associates.

Venables, W.N. (2000). *Exegeses on linear models*. Paper presented to the S-Plus User's Conference, Washington DC, 8-9 October 1998, Washington, DC. Available from: http://www.stats.ox.ac.uk/pub/MASS3/Exegeses.pdf

See Also

Various methods for objects of class afex_aov are available: afex_aov-methods

nice creates the nice ANOVA tables which is by default printed. See also there for a slightly longer discussion of the available effect sizes.

mixed provides a (formula) interface for obtaining p-values for mixed-models via **lme4**. The functions presented here do not estimate mixed models.

Examples

```
#########################
## 1: Specifying ANOVAs ##
#########################
```

```
# Example using a purely within-subjects design
# (Maxwell & Delaney, 2004, Chapter 12, Table 12.5, p. 578):
data(md_12.1)
aov_ez("id", "rt", md_12.1, within = c("angle", "noise"),
       anova_table=list(correction = "none", es = "none"))

# Default output
aov_ez("id", "rt", md_12.1, within = c("angle", "noise"))


# examples using obk.long (see ?obk.long), a long version of the OBrienKaiser dataset (car package).
# Data is a split-plot or mixed design: contains both within- and between-subjects factors.
data(obk.long, package = "afex")

# estimate mixed ANOVA on the full design:
aov_car(value ~ treatment * gender + Error(id/(phase*hour)),
        data = obk.long, observed = "gender")

aov_4(value ~ treatment * gender + (phase*hour|id),
        data = obk.long, observed = "gender")

aov_ez("id", "value", obk.long, between = c("treatment", "gender"),
        within = c("phase", "hour"), observed = "gender")

# the three calls return the same ANOVA table:
# Anova Table (Type 3 tests)
#
# Response: value
#                          Effect          df   MSE         F  ges p.value
# 1                    treatment       2, 10 22.81    3.94 +  .20     .05
# 2                       gender       1, 10 22.81    3.66 +  .11     .08
# 3             treatment:gender       2, 10 22.81      2.86  .18     .10
# 4                        phase 1.60, 15.99  5.02 16.13 ***  .15   .0003
# 5              treatment:phase 3.20, 15.99  5.02    4.85 *  .10     .01
# 6                 gender:phase 1.60, 15.99  5.02      0.28 .003     .71
# 7       treatment:gender:phase 3.20, 15.99  5.02      0.64  .01     .61
# 8                         hour 1.84, 18.41  3.39 16.69 ***  .13  <.0001
# 9               treatment:hour 3.68, 18.41  3.39      0.09 .002     .98
# 10                 gender:hour 1.84, 18.41  3.39      0.45 .004     .63
# 11       treatment:gender:hour 3.68, 18.41  3.39      0.62  .01     .64
# 12                  phase:hour 3.60, 35.96  2.67      1.18  .02     .33
# 13        treatment:phase:hour 7.19, 35.96  2.67      0.35 .009     .93
# 14           gender:phase:hour 3.60, 35.96  2.67      0.93  .01     .45
# 15 treatment:gender:phase:hour 7.19, 35.96  2.67      0.74  .02     .65
#
# Sphericity correction method: GG
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1

# "numeric" variables are per default converted to factors (as long as factorize = TRUE):
obk.long$hour2 <- as.numeric(as.character(obk.long$hour))

# gives same results as calls before
```

```
aov_car(value ~ treatment * gender + Error(id/phase*hour2),
        data = obk.long, observed = c("gender"))


# ANCOVA: adding a covariate (necessary to set factorize = FALSE)
aov_car(value ~ treatment * gender + age + Error(id/(phase*hour)),
        data = obk.long, observed = c("gender", "age"), factorize = FALSE)

aov_4(value ~ treatment * gender + age + (phase*hour|id),
        data = obk.long, observed = c("gender", "age"), factorize = FALSE)

aov_ez("id", "value", obk.long, between = c("treatment", "gender"),
        within = c("phase", "hour"), covariate = "age",
        observed = c("gender", "age"), factorize = FALSE)


# aggregating over one within-subjects factor (phase), with warning:
aov_car(value ~ treatment * gender + Error(id/hour), data = obk.long, observed = "gender")

aov_ez("id", "value", obk.long, c("treatment", "gender"), "hour", observed = "gender")

# aggregating over both within-subjects factors (again with warning),
# only between-subjects factors:
aov_car(value ~ treatment * gender + Error(id), data = obk.long, observed = c("gender"))
aov_4(value ~ treatment * gender + (1|id), data = obk.long, observed = c("gender"))
aov_ez("id", "value", obk.long, between = c("treatment", "gender"), observed = "gender")

# only within-subject factors (ignoring between-subjects factors)
aov_car(value ~ Error(id/(phase*hour)), data = obk.long)
aov_4(value ~ (phase*hour|id), data = obk.long)
aov_ez("id", "value", obk.long, within = c("phase", "hour"))

### changing defaults of ANOVA table:

# no df-correction & partial eta-squared:
aov_car(value ~ treatment * gender + Error(id/(phase*hour)),
        data = obk.long, anova_table = list(correction = "none", es = "pes"))

# no df-correction and no MSE
aov_car(value ~ treatment * gender + Error(id/(phase*hour)),
        data = obk.long,observed = "gender",
        anova_table = list(correction = "none", MSE = FALSE))

# add p-value adjustment for all effects (see Cramer et al., 2015, PB&R)
aov_ez("id", "value", obk.long, between = "treatment",
       within = c("phase", "hour"),
       anova_table = list(p_adjust_method = "holm"))


############################
## 2: Follow-up Analysis ##
############################
```

```
# use data as above
data(obk.long, package = "afex")

# 1. obtain afex_aov object:
a1 <- aov_ez("id", "value", obk.long, between = c("treatment", "gender"),
        within = c("phase", "hour"), observed = "gender")

library("emmeans")  # package emmeans needs to be attached for follow-up tests.

# 1b. plot data (per default with ggplot2):
emmip(a1, gender ~ hour | treatment+phase)

## add univariate CIs:
emmip(a1, gender ~ hour | treatment+phase, CIs = TRUE)

## add multivariate CIs
emmip(a1, gender ~ hour | treatment+phase, CIs = TRUE,
      model = "multivariate")

# use lattice instead of ggplot2 (which has no CIs):
emm_options(graphics.engine = "lattice")
emmip(a1, gender ~ hour | treatment+phase)
emm_options(graphics.engine = "ggplot") # reset options

# 2. obtain reference grid object (default uses univariate model):
r1 <- emmeans(a1, ~treatment +phase)
r1

# multivariate model may be more appropriate
r1 <- emmeans(a1, ~treatment +phase, model = "multivariate")
r1


# 3. create list of contrasts on the reference grid:
c1 <- list(
  A_B_pre = c(rep(0, 6), 0, -1, 1),  # A versus B for pretest
 A_B_comb = c(-0.5, 0.5, 0, -0.5, 0.5, 0, 0, 0, 0), # A vs. B for post and follow-up combined
  effect_post = c(0, 0, 0, -1, 0.5, 0.5, 0, 0, 0), # control versus A&B post
  effect_fup = c(-1, 0.5, 0.5, 0, 0, 0, 0, 0, 0), # control versus A&B follow-up
 effect_comb = c(-0.5, 0.25, 0.25, -0.5, 0.25, 0.25, 0, 0, 0) # control versus A&B combined
)

# 4. test contrasts on reference grid:
contrast(r1, c1)

# same as before, but using Bonferroni-Holm correction for multiple testing:
contrast(r1, c1, adjust = "holm")

# 2. (alternative): all pairwise comparisons of treatment:
emmeans(a1, "treatment", contr = "pairwise", model = "multivariate")

## set multivariate models globally:
# afex_options(emmeans_model = "multivariate")
```

```
#######################
## 3: Other examples ##
#######################
data(obk.long, package = "afex")

# replicating ?Anova using aov_car:
obk_anova <- aov_car(value ~ treatment * gender + Error(id/(phase*hour)),
        data = obk.long, type = 2)
# in contrast to aov you do not need the within-subject factors outside Error()

str(obk_anova, 1, give.attr = FALSE)
# List of 5
#  $ anova_table:Classes 'anova' and 'data.frame': 15 obs. of  6 variables:
#  $ aov        :List of 5
#  $ Anova      :List of 14
#  $ lm         :List of 13
#  $ data       :List of 3

obk_anova$Anova
## Type II Repeated Measures MANOVA Tests: Pillai test statistic
##                             Df test stat approx F num Df den Df       Pr(>F)
## (Intercept)                  1     0.970      318      1     10 0.0000000065 ***
## treatment                    2     0.481        5      2     10      0.03769 *
## gender                       1     0.204        3      1     10      0.14097
## treatment:gender             2     0.364        3      2     10      0.10447
## phase                        1     0.851       26      2      9      0.00019 ***
## treatment:phase              2     0.685        3      4     20      0.06674 .
## gender:phase                 1     0.043        0      2      9      0.82000
## treatment:gender:phase       2     0.311        1      4     20      0.47215
## hour                         1     0.935       25      4      7      0.00030 ***
## treatment:hour               2     0.301        0      8     16      0.92952
## gender:hour                  1     0.293        1      4      7      0.60237
## treatment:gender:hour        2     0.570        1      8     16      0.61319
## phase:hour                   1     0.550        0      8      3      0.83245
## treatment:phase:hour         2     0.664        0     16      8      0.99144
## gender:phase:hour            1     0.695        1      8      3      0.62021
## treatment:gender:phase:hour  2     0.793        0     16      8      0.97237
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`compare.2.vectors` *Compare two vectors using various tests.*

Description

Compares two vectors `x` and `y` using t-test, Welch-test (also known as Satterthwaite), Wilcoxon-test, and a permutation test implemented in **coin**.

Usage

```
compare.2.vectors(x, y, paired = FALSE, na.rm = FALSE,
     tests = c("parametric", "nonparametric"), coin = TRUE,
     alternative = "two.sided",
     perm.distribution,
     wilcox.exact = NULL, wilcox.correct = TRUE)
```

Arguments

x	a (non-empty) numeric vector of data values.
y	a (non-empty) numeric vector of data values.
paired	a logical whether the data is paired. Default is FALSE.
na.rm	logical. Should NA be removed? Default is FALSE.
tests	Which tests to report, parametric or nonparamteric? The default c("parametric","nonparametric") reports both. See details. (Arguments may be abbreviated).
coin	logical or character. Should (permutation) tests from the **coin** package be reported? Default is TRUE corresponding to all implemented tests. FALSE calculates no tests from **coin**. A character vector may include any of the following (potentially abbreviated) implemented tests (see also Details): c("permutation","Wilcoxon","median"
alternative	a character, the alternative hypothesis must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter, will be passed to all functions.
perm.distribution	distribution argument to **coin**, see NullDistribution or , IndependenceTest. If missing, defaults to coin::approximate(100000) indicating an approximation of the excat conditional distribution with 100.000 Monte Carlo samples. One can use "exact" for small samples and if paired = FALSE.
wilcox.exact	exact argument to wilcox.test.
wilcox.correct	correct argument to wilcox.test.

Details

The parametric tests (currently) only contain the *t*-test and Welch/Statterwaithe/Smith/unequal variance *t*-test implemented in t.test. The latter one is only displayed if paired = FALSE.

The nonparametric tests (currently) contain the Wilcoxon test implemented in wilcox.test (stats::Wilcoxon) and (if coin = TRUE) the following tests implemented in **coin**:

- a permutation test oneway_test (the only test in this selction not using a rank transformation),
- the Wilcoxon test wilcox_test (coin::Wilcoxon), and
- the median test median_test.

Note that the two implementations of the Wilcoxon test probably differ. This is due to differences in the calculation of the Null distributions.

Value

a list with up to two elements (i.e., `paramteric` and/or `nonparamteric`) each containing a `data.frame` with the following columns: `test`, `test.statistic`, `test.value`, `test.df`, `p`.

Examples

```
with(sleep, compare.2.vectors(extra[group == 1], extra[group == 2]))

# gives:
## $parametric
##    test test.statistic test.value test.df       p
## 1     t              t     -1.861   18.00 0.07919
## 2 Welch              t     -1.861   17.78 0.07939
##
## $nonparametric
##              test test.statistic test.value test.df       p
## 1 stats::Wilcoxon              W     25.500      NA 0.06933
## 2     permutation              Z     -1.751      NA 0.08154
## 3  coin::Wilcoxon              Z     -1.854      NA 0.06487
## 4          median              Z     -1.744      NA 0.17867

# compare with:
with(sleep, compare.2.vectors(extra[group == 1], extra[group == 2],
                              alternative = "less"))

with(sleep, compare.2.vectors(extra[group == 1], extra[group == 2],
                              alternative = "greater"))

# doesn't make much sense as the data is not paired, but whatever:
with(sleep, compare.2.vectors(extra[group == 1], extra[group == 2],
                              paired = TRUE))

# from ?t.test:
compare.2.vectors(1:10,y=c(7:20, 200))
```

`ems` *Expected values of mean squares for factorial designs Implements the Cornfield-Tukey algorithm for deriving the expected values of the mean squares for factorial designs.*

Description

Expected values of mean squares for factorial designs

Implements the Cornfield-Tukey algorithm for deriving the expected values of the mean squares for factorial designs.

Usage

```
ems(design, nested = NULL, random = "")
```

Arguments

design	A formula object specifying the factors in the design (except residual error, which is always implicitly included). The left hand side of the ~ is the symbol that will be used to denote the number of replications per lowest-level factor combination (I usually use "r" or "n"). The right hand side should include all fixed and random factors separated by *. Factor names should be single letters.
nested	A character vector, where each element is of the form "A/B", indicating that the levels of factor B are nested under the levels of factor A.
random	A character string indicating, without spaces or any separating characters, which of the factors specified in the design are random.

Value

The returned value is a formatted table where the rows represent the mean squares, the columns represent the variance components that comprise the various mean squares, and the entries in each cell represent the terms that are multiplied and summed to form the expectation of the mean square for that row. Each term is either the lower-case version of one of the experimental factors, which indicates the number of levels for that factor, or a "1", which means the variance component for that column is contributes to the mean square but is not multiplied by anything else.

Note

Names for factors or parameters should only be of length 1 as they are simply concatenated in the returned table.

Author(s)

Jake Westfall

See Also

A detailed description with explanation of the example can be found elsewhere (note that the design argument of the function described at the link behaves slightly different).

Example applications of this function can be found here: http://stats.stackexchange.com/a/122662/442.

Examples

```
# 2x2 mixed anova
# A varies between-subjects, B varies within-subjects
ems(r ~ A*B*S, nested="A/S", random="S")

# Clark (1973) example
# random Subjects, random Words, fixed Treatments
ems(r ~ S*W*T, nested="T/W", random="SW")
```

```
# EMSs for Clark design if Words are fixed
ems(r ~ S*W*T, nested="T/W", random="S")
```

`fhch2010` *Data from Freeman, Heathcote, Chalmers, & Hockley (2010)*

Description

Lexical decision and word naming latencies for 300 words and 300 nonwords presented in Freeman, Heathcote, Chalmers, and Hockley (2010). The study had one between-subjects factors, `"task"` with two levels (`"naming"` or `"lexdec"`), and four within-subjects factors: `"stimulus"` type with two levels (`"word"` or `"nonword"`), word `"density"` and word `"frequency"` each with two levels (`"low"` and `"high"`) and stimulus `"length"` with three levels (4, 5, and 6).

Usage

```
fhch2010
```

Format

A `data.frame` with 13,222 obs. of 9 variables:

id participant id, `factor`

task `factor` with two levels indicating which task was performed: `"naming"` or `"lexdec"`

stimulus `factor` indicating whether the shown stimulus was a `"word"` or `"nonword"`

density `factor` indicating the neighborhood density of presented items with two levels: `"low"` and `"high"`. Density is defined as the number of words that differ from a base word by one letter or phoneme.

frequency `factor` indicating the word frequency of presented items with two levels: `"low"` (i.e., words that occur less often in natural language) and `"high"` (i.e., words that occur more often in natural language).

length `factor` with 3 levels (4, 5, or 6) indicating the number of characters of presented stimuli.

item `factor` with 600 levels: 300 words and 300 nonwords

rt response time in seconds

log_rt natural logarithm of response time in seconds

correct boolean indicating whether or not the response in the lexical decision task was correct or incorrect (incorrect responses of the naming task are not part of the data).

Details

In the lexical-decision condition (N = 25), subjects indicated whether each item was a word or a nonword, by pressing either the left (labeled word) or right (labeled nonword) outermost button on a 6-button response pad. The next study item appeared immediately after the lexical decision response was given. In the naming condition (N = 20), subjects were asked to name each item aloud, and items remained on screen for 3 s. Naming time was recorded by a voice key.

Items consisted of 300 words, 75 in each set making up a factorial combination of high and low density and frequency, and 300 nonwords, with equal numbers of 4, 5, and 6 letter items in each set.

Source

Freeman, E., Heathcote, A., Chalmers, K., & Hockley, W. (2010). Item effects in recognition memory for words. Journal of Memory and Language, 62(1), 1-18. http://doi.org/10.1016/j.jml.2009.09.004

Examples

```
data("fhch2010")
str(fhch2010)

a1 <- aov_ez("id", "log_rt", fhch2010, between = "task",
             within = c("density", "frequency", "length", "stimulus"))
nice(a1)

if (requireNamespace("emmeans")) {
  emmeans::emmip(a1, frequency~length|task+stimulus)

  emmeans::emmip(a1, frequency~density|task+stimulus)
}


## Not run:
a2 <- aov_ez("id", "rt", fhch2010, between = "task",
             within = c("density", "frequency", "length", "stimulus"))
nice(a2)

if (requireNamespace("emmeans")) {
  emmeans::emmip(a2, frequency~length|task+stimulus)

  emmeans::emmip(a2, frequency~density|task+stimulus)
}

## End(Not run)
```

---

ks2013.3 *Data from Klauer & Singmann (2013, Experiment 3)*

Description

Klauer and Singmann (2013) attempted to replicate an hypothesis of Morsanyi and Handley (2012) according to which individuals have an intuitive sense of logicality. Specifically, Morsanyi and Handley apparently provided evidence that the logical status of syllogisms (i.e., valid or invalid) affects participants liking ratings of the conclusion of syllogisms. Conclusions from valid syllogisms (e.g., Some snakes are poisonous. No poisonous animals are obbs. Some snakes are not obbs.) received higher liking ratings than conclusions from invalid syllogisms (e.g., No ice creams are vons. Some vons are hot. Some ice creams are not hot.). It is important to noted that in the experiments participants were simply shown the premises and conclusion in succession, they were not asked whether or not the conclusion follows or to generate their own conclusion. Their task was simply to judge how much they liked the "final" statement (i.e., the conclusion).

Usage

```
ks2013.3
```

Format

A data.frame with 1440 rows and 6 variables.

Details

In their Experiment 3 Klauer and Singmann (2013) tested the idea that this finding was a consequence of the materials used and not an effect intuitive logic. More specifically, they observed that in the original study by Morsanyi and Handley (2012) a specific content always appeared with the same logical status. For example, the "ice-cream" content only ever appeared as an invalid syllogism as in the example above but never in a valid syllogism. In other words, content was perfectly confounded with logical status in the original study. To test this they compared a condition in which the logical status was confounded with the content (the "fixed" condition) with a condition in which the contents were randomly assigned to a logical status across participants (the "random" condition). For example, the ice-cream content was, across participants, equally like to appear in the invalid form as given above or in the following valid form: No hot things are vons. Some vons are ice creams. Conclusion Some ice creams are not hot.

The data.frame contains the raw responses of all 60 participants (30 per condition) reported in Klauer & Singmann (2013). Each participants provided 24 responses, 12 to valid and 12 to invalid syllogisms. Furthermore, 8 syllogisms had a believable conclusion (e.g., Some ice creams are not hot.), 8 had an abstract conclusion (e.g., Some snakes are not obbs.), and 8 had an unbelievable conclusion (e.g., Some animals are not monkeys.). The number of the contents corresponds to the numbering given in Morsanyi and Handley (2012, p. 616).

Source

Klauer, K. C., & Singmann, H. (2013). Does logic feel good? Testing for intuitive detection of logicality in syllogistic reasoning. Journal of Experimental Psychology: Learning, Memory, and Cognition, 39(4), 1265-1273. http://doi.org/10.1037/a0030530

Morsanyi, K., & Handley, S. J. (2012). Logic feels so good-I like it! Evidence for intuitive detection of logicality in syllogistic reasoning. Journal of Experimental Psychology: Learning, Memory, and Cognition, 38(3), 596-616. http://doi.org/10.1037/a0026099

Examples

```
data("ks2013.3")

# replicate results reported in Klauer & Singmann (2013, p. 1270)

aov_ez("id", "response", ks2013.3, between = "condition",
       within = c("believability", "validity"))

aov_ez("id", "response", subset(ks2013.3, condition == "fixed"),
       within = c("believability", "validity"))

aov_ez("id", "response", subset(ks2013.3, condition == "random"),
       within = c("believability", "validity"))
```

---

md_12.1 *Data 12.1 from Maxwell & Delaney*

Description

Hypothetical Reaction Time Data for 2 x 3 Perceptual Experiment: Example data for chapter 12 of Maaxwell and Delaney (2004, Table 12.1, p. 574) in long format. Has two within.subjects factors: angle and noise.

Usage

```
md_12.1
```

Format

A data.frame with 60 rows and 4 variables.

Details

Description from pp. 573:

Suppose that a perceptual psychologist studying the visual system was interested in determining the extent to which interfering visual stimuli slow the ability to recognize letters. Subjects are brought into a laboratory and seated in front of a tachistoscope. Subjects are told that they will see either the letter T or the letter I displayed on the screen. In some trials, the letter appears by itself, but in other trials, the target letter is embedded in a group of other letters. This variation in the display constitutes the first factor, which is referred to as noise. The noise factor has two levels?absent and present. The other factor varied by the experimenter is where in the display the target letter appears. This factor, which is called angle, has three levels. The target letter is either shown at the center of the screen (i.e., 0° off-center, where the subject has been instructed to fixate), 4° off-center or 8° off-center (in each case, the deviation from the center varies randomly between left and right). Table 12.1 presents hypothetical data for 10 subjects. As usual, the sample size is kept small to make the calculations easier to follow. The dependent measure is reaction time (latency), measured in milliseconds (ms), required by a subject to identify the correct target letter. Notice that each subject

has six scores, one for each combination of the 2 x 3 design. In an actual perceptual experiment, each of these six scores would itself be the mean score for that subject across a number of trials in the particular condition. Although "trials" could be used as a third within-subjects factor in such a situation, more typically trials are simply averaged over to obtain a more stable measure of the individual's performance in each condition.

Source

Maxwell, S. E., & Delaney, H. D. (2004). Designing experiments and analyzing data: a model-comparisons perspective. Mahwah, N.J.: Lawrence Erlbaum Associates. p. 574

Examples

```
data(md_12.1)

# Table 12.5 (p. 578):
aov_ez("id", "rt", md_12.1, within = c("angle", "noise"),
       args.return=list(correction = "none", es = "none"))
```

---

md_15.1 *Data 15.1 / 11.5 from Maxwell & Delaney*

Description

Hypothetical IQ Data from 12 children at 4 time points: Example data for chapter 11/15 of Maxwell and Delaney (2004, Table 15.1, p. 766) in long format. Has two one within-subjects factor: time.

Usage

```
md_15.1
```

Format

A data.frame with 48 rows and 4 variables.

Details

Description from pp. 534:

The data show that 12 subjects have been observed in each of 4 conditions. To make the example easier to discuss, let's suppose that the 12 subjects are children who have been observed at 30, 36, 42, and 48 months of age. In each case, the dependent variable is the child's age-normed general cognitive score on the McCarthy Scales of Children's Abilities. Although the test is normed so that the mean score is independent of age for the general population, our 12 children may come from a population in which cognitive abilities are either growing more rapidly or less rapidly than average. Indeed, this is the hypothesis our data allow us to address. In other words, although the sample

means suggest that the children's cognitive abilities are growing, a significance test is needed if we want to rule out sampling error as a likely explanation for the observed differences.

To replicate the results in chapter 15 several different contrasts need to be applied, see Examples.

`time` is time in months (centered at 0) and `timecat` is the same as a categorical variable.

Author(s)

R code for examples written by Ulf Mertens and Henrik Singmann

Source

Maxwell, S. E., & Delaney, H. D. (2004). Designing experiments and analyzing data: a model-comparisons perspective. Mahwah, N.J.: Lawrence Erlbaum Associates. p. 766

Examples

```
### replicate results from Table 15.2 to 15.6 (Maxwell & Delaney, 2004, pp. 774)
data(md_15.1)

### ANOVA results (Table 15.2)
aov_4(iq ~ timecat + (timecat|id),data=md_15.1, anova_table=list(correction = "none"))

### Table 15.3 (random intercept only)
# we need to set the base level on the last level:
contrasts(md_15.1$timecat) <- contr.treatment(4, base = 4)
# "Type 3 Tests of Fixed Effects"
(t15.3 <- mixed(iq ~ timecat + (1|id),data=md_15.1, check.contrasts=FALSE))
# "Solution for Fixed Effects" and "Covariance Parameter Estimates"
summary(t15.3$full.model)

### make Figure 15.2
plot(NULL, NULL, ylim = c(80, 140), xlim = c(30, 48), ylab = "iq", xlab = "time")
plyr::d_ply(md_15.1, plyr::.(id), function(x) lines(as.numeric(as.character(x$timecat)), x$iq))

### Table 15.4, page 789
# random intercept plus slope
(t15.4 <- mixed(iq ~ timecat + (1+time|id),data=md_15.1, check.contrasts=FALSE))
summary(t15.4$full.model)

### Table 15.5, page 795
# set up polynomial contrasts for timecat
contrasts(md_15.1$timecat) <- contr.poly
# fit all parameters separately
(t15.5 <- mixed(iq ~ timecat + (1+time|id), data=md_15.1, check.contrasts=FALSE,
                  per.parameter="timecat"))
# quadratic trend is considerably off, conclusions stay the same.


### Table 15.6, page 797
# growth curve model
(t15.6 <- mixed(iq ~ time + (1+time|id),data=md_15.1))
```

```
summary(t15.6$full.model)
```

`md_16.1` *Data 16.1 / 10.9 from Maxwell & Delaney*

Description

Hypothetical Reaction Time Data for 2 x 3 Perceptual Experiment: Example data for chapter 12 of Maaxwell and Delaney (2004, Table 12.1, p. 574) in long format. Has two within.subjects factors: angle and noise.

Usage

```
md_16.1
```

Format

A data.frame with 24 rows and 3 variables.

Details

Description from pp. 829:

As brief background, the goal of the study here is to examine the extent to which female and male clinical psychology graduate student trainees may assign different severity ratings to clients at initial intake. Three female and 3 male graduate students are randomly selected to participate and each is randomly assigned four clients with whom to do an intake interview, after which each clinical trainee assigns a severity rating to each client, producing the data shown in Table 16.1.

Note that I changed the labeling of the id slightly, so that they are now labeled from 1 to 6. Furthermore, I changed the contrasts of sex to `contr.treatment` to replicate the exact results of Table 16.3 (p. 837).

Source

Maxwell, S. E., & Delaney, H. D. (2004). Designing experiments and analyzing data: a model-comparisons perspective. Mahwah, N.J.: Lawrence Erlbaum Associates. p. 574

Examples

```
### replicate results from Table 16.3 (Maxwell & Delaney, 2004, p. 837)
data(md_16.1)

# original results need treatment contrasts:
(mixed1_orig <- mixed(severity ~ sex + (1|id), md_16.1, check.contrasts=FALSE))
summary(mixed1_orig$full.model)

# p-values stay the same with afex default contrasts (contr.sum),
# but estimates and t-values for the fixed effects parameters change.
```

```
(mixed1 <- mixed(severity ~ sex + (1|id), md_16.1))
summary(mixed1$full.model)
```

---

md_16.4 *Data 16.4 from Maxwell & Delaney*

Description

Data from a hypothetical inductive reasoning study.

Usage

```
md_16.4
```

Format

A data.frame with 24 rows and 3 variables.

Details

Description from pp. 841:

Suppose an educational psychologist has developed an intervention to teach inductive reasoning skills to school children. She decides to test the efficacy of her intervention by conducting a randomized design. Three classrooms of students are randomly assigned to the treatment condition, and 3 other classrooms are assigned to the control.

Table 16.4 shows hypothetical data collected from 29 children who participated in the study assessing the effectiveness of the intervention to increase inductive reasoning skills. We want to call your attention to several aspects of the data. First, the 15 children with condition values of 0 received the control, whereas the 14 children with condition values of 1 received the treatment. Second, 4 of the children in the control condition were students in control Classroom 1, 6 of them were students in control Classroom 2, and 5 were students in control Classroom 3. Along similar lines, 3 of the children in the treatment condition were students in treatment Classroom 1, 5 were students in treatment Classroom 2, and 6 were students in treatment Classroom 3. It is essential to understand that there are a total of six classrooms here; we have coded classroom from 1 to 3 for control as well as treatment, because we will indicate to PROC MIXED that classroom is nested under treatment. Third, scores on the dependent variable appear in the rightmost column under the variable label "induct."

Note that it would make a lot more sense to change the labeling of room from 1 to 3 nested within cond to 1 to 6. However, I keep this in line with the original. The random effects term in the call to mixed is therefore a little bit uncommon.#'

Source

Maxwell, S. E., & Delaney, H. D. (2004). Designing experiments and analyzing data: a model-comparisons perspective. Mahwah, N.J.: Lawrence Erlbaum Associates. p. 574

Examples

```
# data for next examples (Maxwell & Delaney, Table 16.4)
data(md_16.4)
str(md_16.4)

### replicate results from Table 16.6 (Maxwell & Delaney, 2004, p. 845)
# p-values (almost) hold:
(mixed2 <- mixed(induct ~ cond + (1|room:cond), md_16.4))
# (1|room:cond) is needed because room is nested within cond.
```

---

mixed *p-values for fixed effects of mixed-model via lme4::lmer()*

Description

Estimates mixed models with **lme4** and calculates p-values for all fixed effects. The default method `"KR"` (= Kenward-Roger) as well as `method="S"` (Satterthwaite) support LMMs and estimate the model with `lmer` and then pass it to the `lmerTest` anova method (or `Anova`). The other methods (`"LRT"` = likelihood-ratio tests and `"PB"` = parametric bootstrap) support both LMMs (estimated via `lmer`) and GLMMs (i.e., with `family` argument which invokes estimation via `glmer`) and estimate a full model and restricted models in which the parameters corresponding to one effect (i.e., model term) are withhold (i.e., fixed to 0). Per default tests are based on Type 3 sums of squares. `print`, `nice`, `anova`, and `summary` methods for the returned object of class `"mixed"` are available. `summary` invokes the default **lme4** summary method and shows parameters instead of effects.

`lmer_alt` is simply a wrapper for mixed that only returns the `"lmerModLmerTest"` or `"merMod"` object and correctly uses the `||` notation for removing correlations among factors. This function otherwise behaves like `g/lmer` (as for `mixed`, it calls `glmer` as soon as a `family` argument is present). Use `afex_options("lmer_function")` to set which function for estimation should be used. This option determines the class of the returned object (i.e., `"lmerModLmerTest"` or `"merMod"`).

Usage

```
mixed(formula, data, type = afex_options("type"),
  method = afex_options("method_mixed"), per_parameter = NULL,
  args_test = list(), test_intercept = FALSE,
  check_contrasts = afex_options("check_contrasts"), expand_re = FALSE,
  all_fit = FALSE, set_data_arg = afex_options("set_data_arg"),
  progress = TRUE, cl = NULL, return = "mixed",
  sig_symbols = afex_options("sig_symbols"), ...)

lmer_alt(formula, data, check_contrasts = FALSE, ...)
```

Arguments

formula	a formula describing the full mixed-model to be fitted. As this formula is passed to lmer, it needs at least one random term.
data	data.frame containing the data. Should have all the variables present in fixed, random, and dv as columns.
type	type of test on which effects are based. Default is to use type 3 tests, taken from afex_options.
method	character vector indicating which methods for obtaining p-values should be used: "KR" corresponds to the Kenward-Roger approximation for degrees of freedom (only LMMs), "S" corresponds to the Satterthwaite approximation for degrees of freedom (via lmerTest, only LMMs), "PB" calculates p-values based on parametric bootstrap, "LRT" calculates p-values via the likelihood ratio tests implemented in the anova method for merMod objects (only recommended for models with many [i.e., > 50] levels for the random factors). The default (currently "KR") is taken from afex_options. For historical compatibility "nested-KR" is also supported which was the default KR-method in previous versions.
per_parameter	character vector specifying for which variable tests should be run for each parameter (instead for the overall effect). Can be useful e.g., for testing ordered factors. Uses grep for selecting parameters among the fixed effects so regular expressions (regex) are possible. See Examples.
args_test	list of arguments passed to the function calculating the p-values. See Details.
test_intercept	logical. Whether or not the intercept should also be fitted and tested for significance. Default is FALSE. Only relevant if type = 3.
check_contrasts	logical. Should contrasts be checked and (if necessary) changed to "contr.sum"? See Details. The default ("TRUE") is taken from afex_options.
expand_re	logical. Should random effects terms be expanded (i.e., factors transformed into numerical variables) before fitting with (g)lmer? Allows to use "||" notation with factors.
all_fit	logical. Should all_fit be used to fit each model with each available optimization algorithm and the results that provided the best fit in each case be used? Warning: This can dramatically increase the optimization time. Adds two new attributes to the returned object designating which algorithm was selected and the log-likelihoods for each algorithm. Note that only warnings from the initial fit are emitted during fitting. The warnings of the chosen models are emitted when printing the returned object.
set_data_arg	logical. Should the data argument in the slot call of the merMod object returned from lmer be set to the passed data argument? If FALSE (currently the default) the name will be data. TRUE may be helpful when fitted objects are used afterwards (e.g., compared using anova or when using the effects package, see examples). **emmeans** functions appear to work better with FALSE. Default is given by afex_options("set_data_arg").
progress	if TRUE, shows progress with a text progress bar and other status messages during estimation

`cl` A vector identifying a cluster; used for distributing the estimation of the different models using several cores (if seveal models are calculated). See examples. If `ckeck_contrasts = TRUE`, mixed sets the current contrasts (`getOption("contrasts")`) at the nodes. Note this does *not* distribute calculation of p-values (e.g., when using `method = "PB"`) across the cluster. Use `args_test` for this.

`return` the default is to return an object of class `"mixed"`. `return = "merMod"` will skip the calculation of all submodels and p-values and simply return the full model estimated with `lmer` (note that somewhat unintuiviely, the returned object can either be of class `"lmerModLmerTest"` or of class `"merMod"`, depending on the value of `afex_options("lmer_function")`). Can be useful in combination with `expand_re = TRUE` which allows to use "||" with factors. `return = "data"` will not fit any models but just return the data that would have been used for estimating the model (note that the data is also part of the returned object).

`sig_symbols` Character. What should be the symbols designating significance? When entering an vector with `length(sig.symbol) < 4` only those elements of the default (`c(" +", " *", " **", " ***")`) will be replaced. `sig_symbols = ""` will display the stars but not the +, `sig_symbols = rep("", 4)` will display no symbols. The default is given by `afex_options("sig_symbols")`.

`...` further arguments (such as `weights`, `family`, or `control`) passed to `lmer`/`glmer`. Note that additional data (e.g., `weights`) need to be passed fully and not only by name (e.g., `weights = df$weights` and not `weights = weights`).

Details

For an introduction to mixed-modeling for experimental designs see our chapter (Singmann & Kellen, in press) or Barr, Levy, Scheepers, & Tily (2013). Arguments for using the Kenward-Roger approximation for obtaining p-values are given by Judd, Westfall, and Kenny (2012). Further introductions to mixed-modeling for experimental designs are given by Baayen and colleagues (Baayen, 2008; Baayen, Davidson & Bates, 2008; Baayen & Milin, 2010). Specific recommendations on which random effects structure to specify for confirmatory tests can be found in Barr and colleagues (2013) and Barr (2013), but also see Bates et al. (2015).

p-value Calculations:

When `method = "KR"` (the default, implemented via `KRmodcomp`), the Kenward-Roger approximation for degrees-of-freedom is calculated using `lmerTest` (if `test_intercept=FALSE`) or `Anova` (if `test_intercept=TRUE`), which is only applicable to linear-mixed models (LMMs). The test statistic in the output is an F-value (`F`). A similar method that requires less RAM is `method = "S"` which calculates the Satterthwaite approximation for degrees-of-freedom via `lmerTest` and is also only applicable to LMMs. `method = "KR"` or `method = "S"` provide the best control for Type 1 errors for LMMs (Luke, 2017).

`method = "PB"` calculates p-values using parametric bootstrap using `PBmodcomp`. This can be used for linear and also generalized linear mixed models (GLMMs) by specifying a `family` argument to `mixed`. Note that you should specify further arguments to `PBmodcomp` via `args_test`, especially `nsim` (the number of simulations to form the reference distribution) or `cl` (for using multiple cores). For other arguments see `PBmodcomp`. Note that `REML` (argument to `[g]lmer`) will be set to `FALSE` if method is `PB`.

`method = "LRT"` calculates p-values via likelihood ratio tests implemented in the `anova` method for `"merMod"` objects. This is the method recommended by Barr et al. (2013; which did not

test the other methods implemented here). Using likelihood ratio tests is only recommended for models with many levels for the random effects (> 50), but can be pretty helpful in case the other methods fail (due to memory and/or time limitations). The lme4 faq also recommends the other methods over likelihood ratio tests.

Implementation Details:

For methods "KR" and "S" type 3 and 2 tests are implemented as in Anova.

For all other methods, type 3 tests are obtained by comparing a model in which only the tested effect is excluded with the full model (containing all effects). For method "nested-KR" (which was the default in previous versions) this corresponds to the (type 3) Wald tests given by car::Anova for "lmerMod" models. The submodels in which the tested effect is excluded are obtained by manually creating a model matrix which is then fitted in "lme4".

Type 2 tests are truly sequential. They are obtained by comparing a model in which the tested effect and all higher oder effect (e.g., all three-way interactions for testing a two-way interaction) are excluded with a model in which only effects up to the order of the tested effect are present and all higher order effects absent. In other words, there are multiple full models, one for each order of effects. Consequently, the results for lower order effects are identical of whether or not higher order effects are part of the model or not. This latter feature is not consistent with classical ANOVA type 2 tests but a consequence of the sequential tests (and I didn't find a better way of implementing the Type 2 tests). This **does not** correspond to the (type 2) Wald test reported by car::Anova.

If check_contrasts = TRUE, contrasts will be set to "contr.sum" for all factors in the formula if default contrasts are not equal to "contr.sum" or attrib(factor,"contrasts") != "contr.sum". Furthermore, the current contrasts (obtained via getOption("contrasts")) will be set at the cluster nodes if cl is not NULL.

Expand Random Effects: expand_re = TRUE allows to expand the random effects structure before passing it to lmer. This allows to disable estimation of correlation among random effects for random effects term containing factors using the || notation which may aid in achieving model convergence (see Bates et al., 2015). This is achieved by first creating a model matrix for each random effects term individually, rename and append the so created columns to the data that will be fitted, replace the actual random effects term with the so created variables (concatenated with +), and then fit the model. The variables are renamed by prepending all variables with rei (where i is the number of the random effects term) and replacing ":" with "_by_".

lmer_alt is simply a wrapper for mixed that is intended to behave like lmer (or glmer if a family argument is present), but also allows the use of || with factors (by always using expand_re = TRUE). This means that lmer_alt per default does not enforce a specific contrast on factors and only returns the "lmerModLmerTest" or "merMod" object without calculating any additional models or p-values (this is achieved by setting return = "merMod"). Note that it most likely differs from g/lmer in how it handles missing values so it is recommended to only pass data without missing values to it!

One consequence of using expand_re = TRUE is that the data that is fitted will not be the same as the passed data.frame which can lead to problems with e.g., the predict method. However, the actual data used for fitting is also returned as part of the mixed object so can be used from there. Note that the set_data_arg can be used to change whether the data argument in the call to g/lmer is set to data (the default) or the name of the data argument passed by the user.

Value

An object of class "mixed" (i.e., a list) with the following elements:

1. anova_table a data.frame containing the statistics returned from KRmodcomp. The stat column in this data.frame gives the value of the test statistic, an F-value for method = "KR" and a chi-square value for the other two methods.
2. full_model the "lmerModLmerTest" or "merMod" object returned from estimating the full model. Use afex_options("lmer_function") for setting which function for estimation should be used. The possible options are "lmerTest" (the default returning an object of class "lmerModLmerTest") and "lme4" returning an object of class ("merMod"). Note that in case a family argument is present an object of class "glmerMod" is always returned.
3. restricted_models a list of "g/lmerMod" (or "lmerModLmerTest") objects from estimating the restricted models (i.e., each model lacks the corresponding effect)
4. tests a list of objects returned by the function for obtaining the p-values.
5. data The data used for estimation (i.e., after excluding missing rows and applying expand_re if requested).
6. call The matched call.

It also has the following attributes, "type" and "method". And the attributes "all_fit_selected" and "all_fit_logLik" if all_fit=TRUE.

Two similar methods exist for objects of class "mixed": print and anova. They print a nice version of the anova_table element of the returned object (which is also invisibly returned). This methods omit some columns and nicely round the other columns. The following columns are always printed:

1. Effect name of effect
2. p.value estimated p-value for the effect

For LMMs with method="KR" or method="S" the following further columns are returned (note: the Kenward-Roger correction does two separate things: (1) it computes an effective number for the denominator df; (2) it scales the statistic by a calculated amount, see also http://stackoverflow.com/a/25612960/289572):

1. F computed F statistic
2. ndf numerator degrees of freedom (number of parameters used for the effect)
3. ddf denominator degrees of freedom (effective residual degrees of freedom for testing the effect), computed from the Kenward-Roger correction using pbkrtest::KRmodcomp
4. F.scaling scaling of F-statistic computing from Kenward-Roger approximation (only printed if method="nested-KR")

For models with method="LRT" the following further columns are returned:

1. df.large degrees of freedom (i.e., estimated paramaters) for full model (i.e., model containing the corresponding effect)
2. df.small degrees of freedom (i.e., estimated paramaters) for restricted model (i.e., model without the corresponding effect)
3. chisq 2 times the difference in likelihood (obtained with logLik) between full and restricted model

4. df difference in degrees of freedom between full and restricted model (p-value is based on these df).

For models with method="PB" the following further column is returned:

1. stat 2 times the difference in likelihood (obtained with logLik) between full and restricted model (i.e., a chi-square value).

Note that anova can also be called with additional mixed and/or merMod objects. In this casethe full models are passed on to anova.merMod (with refit=FALSE, which differs from the default of anova.merMod) which produces the known LRT tables.

The summary method for objects of class mixed simply calls summary.merMod on the full model.

If return = "merMod" (or when invoking lmer_alt), an object of class "lmerModLmerTest" or of class "merMod" (depending on the value of afex_options("lmer_function")), as returned from g/lmer, is returned. The default behavior is to return an object of class "lmerModLmerTest" estimated via lmer.

Note

When method = "KR", obtaining p-values is known to crash due too insufficient memory or other computational limitations (especially with complex random effects structures). In these cases, the other methods should be used. The RAM demand is a problem especially on 32 bit Windows which only supports up to 2 or 3GB RAM (see R Windows FAQ). Then it is probably a good idea to use methods "S", "LRT", or "PB".

"mixed" will throw a message if numerical variables are not centered on 0, as main effects (of other variables then the numeric one) can be hard to interpret if numerical variables appear in interactions. See Dalal & Zickar (2012).

Per default mixed uses lmer, this can be changed to lmer by calling: afex_options(lmer_function = "lme4")

Formulas longer than 500 characters will most likely fail due to the use of deparse.

Please report bugs or unexpected behavior by opening a guthub issue: https://github.com/singmann/afex/issues

Author(s)

Henrik Singmann with contributions from Ben Bolker and Joshua Wiley.

References

Baayen, R. H. (2008). *Analyzing linguistic data: a practical introduction to statistics using R*. Cambridge, UK; New York: Cambridge University Press.

Baayen, R. H., Davidson, D. J., & Bates, D. M. (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59(4), 390-412. doi:10.1016/j.jml.2007.12.005

Baayen, R. H., & Milin, P. (2010). Analyzing Reaction Times. *International Journal of Psychological Research*, 3(2), 12-28.

Barr, D. J. (2013). Random effects structure for testing interactions in linear mixed-effects models. *Frontiers in Quantitative Psychology and Measurement*, 328. doi:10.3389/fpsyg.2013.00328

Barr, D. J., Levy, R., Scheepers, C., & Tily, H. J. (2013). Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, 68(3), 255-278. doi:10.1016/j.jml.2012.11.001

Bates, D., Kliegl, R., Vasishth, S., & Baayen, H. (2015). *Parsimonious Mixed Models*. arXiv:1506.04967 [stat]. Retrieved from http://arxiv.org/abs/1506.04967

Dalal, D. K., & Zickar, M. J. (2012). Some Common Myths About Centering Predictor Variables in Moderated Multiple Regression and Polynomial Regression. *Organizational Research Methods*, 15(3), 339-362. doi:10.1177/1094428111430540

Judd, C. M., Westfall, J., & Kenny, D. A. (2012). Treating stimuli as a random factor in social psychology: A new and comprehensive solution to a pervasive but largely ignored problem. *Journal of Personality and Social Psychology*, 103(1), 54-69. doi:10.1037/a0028347

Luke, S. (2017). Evaluating significance in linear mixed-effects models in R. *Behavior Research Methods*. https://doi.org/10.3758/s13428-016-0809-y

Maxwell, S. E., & Delaney, H. D. (2004). *Designing experiments and analyzing data: a model-comparisons perspective.* Mahwah, N.J.: Lawrence Erlbaum Associates.

See Also

aov_ez and aov_car for convenience functions to analyze experimental deisgns with classical ANOVA or ANCOVA wrapping Anova.

see the following for the data sets from Maxwell and Delaney (2004) used and more examples: md_15.1, md_16.1, and md_16.4.

Examples

```
##################################
## Simple Examples (from MEMSS) ##
##################################

data("Machines", package = "MEMSS")

# simple model with random-slopes for repeated-measures factor
m1 <- mixed(score ~ Machine + (Machine|Worker), data=Machines)
m1

# suppress correlation among random effect parameters with expand_re = TRUE
m2 <- mixed(score ~ Machine + (Machine||Worker), data=Machines, expand_re = TRUE)
m2

## compare:
summary(m1)$varcor
summary(m2)$varcor
# for wrong solution see:
# summary(lmer(score ~ Machine + (Machine||Worker), data=Machines))$varcor

# follow-up tests
library("emmeans")  # package emmeans needs to be attached for follow-up tests.
(emm1 <- emmeans(m1, "Machine"))
```

```
pairs(emm1, adjust = "holm") # all pairwise comparisons
con1 <- list(
  c1 = c(1, -0.5, -0.5), # 1 versus other 2
  c2 = c(0.5, -1, 0.5) # 1 and 3 versus  2
)
contrast(emm1, con1, adjust = "holm")

# plotting
emmip(m1, ~Machine, CIs = TRUE)
emmip(m2, ~Machine, CIs = TRUE)


## Not run:
########################
### Further Options ###
########################

## Multicore:

require(parallel)
(nc <- detectCores()) # number of cores
cl <- makeCluster(rep("localhost", nc)) # make cluster
# to keep track of what the function is doindg redirect output to outfile:
# cl <- makeCluster(rep("localhost", nc), outfile = "cl.log.txt")

data("Machines", package = "MEMSS")
## There are two ways to use multicore:

# 1. Obtain fits with multicore:
mixed(score ~ Machine + (Machine|Worker), data=Machines, cl = cl)

# 2. Obtain PB samples via multicore:
mixed(score ~ Machine + (Machine|Worker), data=Machines,
 method = "PB", args_test = list(nsim = 50, cl = cl)) # better use 500 or 1000

## Both ways can be combined:
# 2. Obtain PB samples via multicore:
mixed(score ~ Machine + (Machine|Worker), data=Machines, cl = cl,
 method = "PB", args_test = list(nsim = 50, cl = cl))

#### use all_fit = TRUE and expand_re = TRUE:
data("sk2011.2") # data described in more detail below
sk2_aff <- droplevels(sk2011.2[sk2011.2$what == "affirmation",])

require(optimx) # uses two more algorithms
sk2_aff_b <- mixed(response ~ instruction*type+(inference*type||id), sk2_aff,
               expand_re = TRUE, all_fit = TRUE)
attr(sk2_aff_b, "all_fit_selected")
attr(sk2_aff_b, "all_fit_logLik")

# considerably faster with multicore:
clusterEvalQ(cl, library(optimx)) # need to load optimx in cluster
sk2_aff_b2 <- mixed(response ~ instruction*type+(inference*type||id), sk2_aff,
```

```
               expand_re = TRUE, all_fit = TRUE, cl=cl)
attr(sk2_aff_b2, "all_fit_selected")
attr(sk2_aff_b2, "all_fit_logLik")


stopCluster(cl)


## End(Not run)

#####################################################
## Replicating Maxwell & Delaney (2004) Examples ##
#####################################################
## Not run:

### replicate results from Table 15.4 (Maxwell & Delaney, 2004, p. 789)
data(md_15.1)
# random intercept plus random slope
(t15.4a <- mixed(iq ~ timecat + (1+time|id),data=md_15.1))

# to also replicate exact parameters use treatment.contrasts and the last level as base level:
contrasts(md_15.1$timecat) <- contr.treatment(4, base = 4)
(t15.4b <- mixed(iq ~ timecat + (1+time|id),data=md_15.1, check_contrasts=FALSE))
summary(t15.4a)  # gives "wrong" parameters extimates
summary(t15.4b)  # identical parameters estimates

# for more examples from chapter 15 see ?md_15.1

### replicate results from Table 16.3 (Maxwell & Delaney, 2004, p. 837)
data(md_16.1)

# original results need treatment contrasts:
(mixed1_orig <- mixed(severity ~ sex + (1|id), md_16.1, check_contrasts=FALSE))
summary(mixed1_orig$full_model)

# p-value stays the same with afex default contrasts (contr.sum),
# but estimates and t-values for the fixed effects parameters change.
(mixed1 <- mixed(severity ~ sex + (1|id), md_16.1))
summary(mixed1$full_model)


# data for next examples (Maxwell & Delaney, Table 16.4)
data(md_16.4)
str(md_16.4)

### replicate results from Table 16.6 (Maxwell & Delaney, 2004, p. 845)
# Note that (1|room:cond) is needed because room is nested within cond.
# p-value (almost) holds.
(mixed2 <- mixed(induct ~ cond + (1|room:cond), md_16.4))
# (differences are dut to the use of Kenward-Roger approximation here,
# whereas M&W's p-values are based on uncorrected df.)

# again, to obtain identical parameter and t-values, use treatment contrasts:
```

```
summary(mixed2) # not identical

# prepare new data.frame with contrasts:
md_16.4b <- within(md_16.4, cond <- C(cond, contr.treatment, base = 2))
str(md_16.4b)

# p-value stays identical:
(mixed2_orig <- mixed(induct ~ cond + (1|room:cond), md_16.4b, check_contrasts=FALSE))
summary(mixed2_orig$full_model) # replicates parameters


### replicate results from Table 16.7 (Maxwell & Delaney, 2004, p. 851)
# F-values (almost) hold, p-values (especially for skill) are off
(mixed3 <- mixed(induct ~ cond + skill + (1|room:cond), md_16.4))

# however, parameters are perfectly recovered when using the original contrasts:
mixed3_orig <- mixed(induct ~ cond + skill + (1|room:cond), md_16.4b, check_contrasts=FALSE)
summary(mixed3_orig)


### replicate results from Table 16.10 (Maxwell & Delaney, 2004, p. 862)
# for this we need to center cog:
md_16.4b$cog <- scale(md_16.4b$cog, scale=FALSE)

# F-values and p-values are relatively off:
(mixed4 <- mixed(induct ~ cond*cog + (cog|room:cond), md_16.4b))
# contrast has a relatively important influence on cog
(mixed4_orig <- mixed(induct ~ cond*cog + (cog|room:cond), md_16.4b, check_contrasts=FALSE))

# parameters are again almost perfectly recovered:
summary(mixed4_orig)

## End(Not run)

############################
## Full Analysis Example ##
############################

## Not run:
### split-plot experiment (Singmann & Klauer, 2011, Exp. 2)
## between-factor: instruction
## within-factor: inference & type
## hypothesis: three-way interaction
data("sk2011.2")

# use only affirmation problems (S&K also splitted the data like this)
sk2_aff <- droplevels(sk2011.2[sk2011.2$what == "affirmation",])

# set up model with maximal by-participant random slopes
sk_m1 <- mixed(response ~ instruction*inference*type+(inference*type|id), sk2_aff)

sk_m1 # prints ANOVA table with nicely rounded numbers (i.e., as characters)
nice(sk_m1)  # returns the same but without printing potential warnings
```

```
anova(sk_m1) # returns and prints numeric ANOVA table (i.e., not-rounded)
summary(sk_m1) # lmer summary of full model

# same model but using Satterthwaite approximation of df
# very similar results but faster
sk_m1b <- mixed(response ~ instruction*inference*type+(inference*type|id),
                sk2_aff, method="S")
nice(sk_m1b)
# identical results as:
anova(sk_m1$full_model)

# suppressing correlation among random slopes:
# very similar results, but significantly faster and often less convergence warnings.
sk_m2 <- mixed(response ~ instruction*inference*type+(inference*type||id), sk2_aff,
               expand_re = TRUE)
sk_m2

## mixed objects can be passed to emmeans
library("emmeans")  # however, package emmeans needs to be attached first

# recreates basically Figure 4 (S&K, 2011, upper panel)
# only the 4th and 6th x-axis position are flipped
emmip(sk_m1, instruction~type+inference)

# use lattice instead of ggplot2:
emm_options(graphics.engine = "lattice")
emmip(sk_m1, instruction~type+inference)
emm_options(graphics.engine = "ggplot") # reset options

# set up reference grid for custom contrasts:
# this can be made faster via:
emm_options(lmer.df = "Kenward-Roger") # set df for emmeans to KR
# emm_options(lmer.df = "Satterthwaite") # the default
# emm_options(lmer.df = "asymptotic") # the fastest, no df
(rg1 <- emmeans(sk_m1, c("instruction", "type", "inference")))

# set up contrasts on reference grid:
contr_sk2 <- list(
  ded_validity_effect = c(rep(0, 4), 1, rep(0, 5), -1, 0),
  ind_validity_effect = c(rep(0, 5), 1, rep(0, 5), -1),
  counter_MP = c(rep(0, 4), 1, -1, rep(0, 6)),
  counter_AC = c(rep(0, 10), 1, -1)
)

# test the main double dissociation (see S&K, p. 268)
contrast(rg1, contr_sk2, adjust = "holm")
# only plausibility effect is not significant here.

## End(Not run)

####################
## Other Examples ##
####################
```

```
## Not run:

# use the obk.long data (not reasonable, no random slopes)
data(obk.long)
mixed(value ~ treatment * phase + (1|id), obk.long)

# Examples for using the per.parameter argument
# note, require method = "nested-KR", "LRT", or "PB"
# also we use custom contrasts
data(obk.long, package = "afex")
obk.long$hour <- ordered(obk.long$hour)
contrasts(obk.long$phase) <- "contr.sum"
contrasts(obk.long$treatment) <- "contr.sum"

# tests only the main effect parameters of hour individually per parameter.
mixed(value ~ treatment*phase*hour +(1|id), per_parameter = "^hour$",
      data = obk.long, method = "nested-KR", check_contrasts = FALSE)

# tests all parameters including hour individually
mixed(value ~ treatment*phase*hour +(1|id), per_parameter = "hour",
      data = obk.long, method = "nested-KR", check_contrasts = FALSE)

# tests all parameters individually
mixed(value ~ treatment*phase*hour +(1|id), per_parameter = ".",
      data = obk.long, method = "nested-KR", check_contrasts = FALSE)

# example data from package languageR:
# Lexical decision latencies elicited from 21 subjects for 79 English concrete nouns,
# with variables linked to subject or word.
data(lexdec, package = "languageR")

# using the simplest model
m1 <- mixed(RT ~ Correct + Trial + PrevType * meanWeight +
    Frequency + NativeLanguage * Length + (1|Subject) + (1|Word), data = lexdec)
m1
# Mixed Model Anova Table (Type 3 tests, KR-method)
#
# Model: RT ~ Correct + Trial + PrevType * meanWeight + Frequency + NativeLanguage *
# Model:     Length + (1 | Subject) + (1 | Word)
# Data: lexdec
#                   Effect         df         F p.value
# 1               Correct 1, 1627.73   8.15 **    .004
# 2                 Trial 1, 1592.43   7.57 **    .006
# 3              PrevType 1, 1605.39      0.17     .68
# 4            meanWeight   1, 75.39 14.85 ***   .0002
# 5             Frequency   1, 76.08 56.53 ***  <.0001
# 6        NativeLanguage   1, 27.11      0.70     .41
# 7                Length   1, 75.83   8.70 **    .004
# 8   PrevType:meanWeight 1, 1601.18    6.18 *     .01
# 9 NativeLanguage:Length 1, 1555.49 14.24 ***   .0002
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1
```

```
# Fitting a GLMM using parametric bootstrap:
require("mlmRev") # for the data, see ?Contraception

gm1 <- mixed(use ~ age + I(age^2) + urban + livch + (1 | district), method = "PB",
 family = binomial, data = Contraception, args_test = list(nsim = 10))
## note that nsim = 10 is way too low for all real examples!


## End(Not run)

## Not run:
####################################
## Interplay with effects packages ##
####################################

data("Machines", package = "MEMSS")
# simple model with random-slopes for repeated-measures factor
m1 <- mixed(score ~ Machine + (Machine|Worker), data=Machines,
            set_data_arg = TRUE) ## necessary for it to work!

library("effects")

Effect("Machine", m1$full_model) # not correct:
#  Machine effect
# Machine
#        A        B        C
# 59.65000 52.35556 60.32222

# compare:
emmeans::emmeans(m1, "Machine")
 # Machine   emmean       SE  df asymp.LCL asymp.UCL
 # A       52.35556 1.680711 Inf  49.06142  55.64969
 # B       60.32222 3.528546 Inf  53.40640  67.23804
 # C       66.27222 1.806273 Inf  62.73199  69.81245

## necessary to set contr.sum globally:
set_sum_contrasts()
Effect("Machine", m1$full_model)
#  Machine effect
# Machine
#        A        B        C
# 52.35556 60.32222 66.27222

plot(Effect("Machine", m1$full_model))

## End(Not run)
```

---

nice *Make nice ANOVA table for printing.*

Description

This generic function produces a nice ANOVA table for printing for objects of class. `nice_anova` takes an object from `Anova` possible created by the convenience functions `aov_ez` or `aov_car`. When within-subject factors are present, either sphericity corrected or uncorrected degrees of freedom can be reported.

Usage

```
nice(object, ...)

## S3 method for class 'afex_aov'
nice(object, es = attr(object$anova_table, "es"),
  observed = attr(object$anova_table, "observed"),
  correction = attr(object$anova_table, "correction"), MSE = NULL,
  intercept = NULL, p_adjust_method = attr(object$anova_table,
  "p_adjust_method"), sig_symbols = attr(object$anova_table,
  "sig_symbols"), ...)

## S3 method for class 'anova'
nice(object, MSE = NULL, intercept = NULL,
  sig_symbols = attr(object, "sig_symbols"), sig.symbols, ...)

## S3 method for class 'mixed'
nice(object, sig_symbols = attr(object$anova_table,
  "sig_symbols"), ...)

## S3 method for class 'nice_table'
print(x, ...)
```

Arguments

`object, x`	An object of class `"afex_aov"` (see `aov_car`) or of class `"mixed"` (see `mixed`) as returned from the **afex** functions. Alternatively, an object of class `"Anova.mlm"` or `"anova"` as returned from `Anova`.
`...`	currently ignored.
`es`	Effect Size to be reported. The default is given by `afex_options("es_aov")`, which is initially set to `"ges"` (i.e., reporting generalized eta-squared, see details). Also supported is partial eta-squared (`"pes"`) or `"none"`.
`observed`	character vector referring to the observed (i.e., non manipulated) variables/effects in the design. Important for calculation of generalized eta-squared (ignored if `es` is not `"ges"`), see details.
`correction`	Character. Which sphericity correction of the degrees of freedom should be reported for the within-subject factors. The default is given by `afex_options("correction_aov")`, which is initially set to `"GG"` corresponding to the Greenhouse-Geisser correction. Possible values are `"GG"`, `"HF"` (i.e., Hyunh-Feldt correction), and `"none"` (i.e., no correction).

MSE	logical. Should the column containing the Mean Sqaured Error (MSE) be displayed? Default is TRUE.
intercept	logical. Should intercept (if present) be included in the ANOVA table? Default is FALSE which hides the intercept.
p_adjust_method	character indicating if p-values for individual effects should be adjusted for multiple comparisons (see p.adjust and details). The default NULL corresponds to no adjustment.
sig_symbols	Character. What should be the symbols designating significance? When entering an vector with length(sig.symbol) < 4 only those elements of the default (c(" +", " *", " **", " ***")) will be replaced. sig_symbols = "" will display the stars but not the +, sig_symbols = rep("",4) will display no symbols. The default is given by afex_options("sig_symbols").
sig.symbols	deprecated argument, only for backwards compatibility, use "sig_symbols" instead.

Details

The returned data.frame is print-ready when adding to a document with proper methods. Either directly via **knitr** or similar approaches such as via package **xtable** (nowadays **knitr** is probably the best approach, see here). **xtable** converts a data.frame into LaTeX code with many possible options (e.g., allowing for "longtable" or "sidewaystable"), see xtable and print.xtable. See Examples.

Conversion functions to other formats (such as HTML, ODF, or Word) can be found at the Reproducible Research Task View.

The default reports generalized eta squared (Olejnik & Algina, 2003), the "recommended effect size for repeated measured designs" (Bakeman, 2005). Note that it is important that all measured variables (as opposed to experimentally manipulated variables), such as e.g., age, gender, weight, ..., must be declared via observed to obtain the correct effect size estimate. Partial eta squared ("pes") does not require this.

Exploratory ANOVA, for which no detailed hypotheses have been specified a priori, harbor a multiple comparison problem (Cramer et al., 2015). To avoid an inflation of familywise Type I error rate, results need to be corrected for multiple comparisons using p_adjust_method. p_adjust_method defaults to the method specified in the call to aov_car in anova_table. If no method was specified and p_adjust_method = NULL p-values are not adjusted.

Value

A data.frame of class nice_table with the ANOVA table consisting of characters. The columns that are always present are: Effect, df (degrees of freedom), F, and p.

ges contains the generalized eta-squared effect size measure (Bakeman, 2005), pes contains partial eta-squared (if requested).

Author(s)

The code for calculating generalized eta-squared was written by Mike Lawrence.
Everything else was written by Henrik Singmann.

References

Bakeman, R. (2005). Recommended effect size statistics for repeated measures designs. *Behavior Research Methods*, 37(3), 379-384. doi:10.3758/BF03192707

Cramer, A. O. J., van Ravenzwaaij, D., Matzke, D., Steingroever, H., Wetzels, R., Grasman, R. P. P. P., ... Wagenmakers, E.-J. (2015). Hidden multiplicity in exploratory multiway ANOVA: Prevalence and remedies. *Psychonomic Bulletin & Review*, 1-8. doi:10.3758/s13423-015-0913-5

Olejnik, S., & Algina, J. (2003). Generalized Eta and Omega Squared Statistics: Measures of Effect Size for Some Common Research Designs. *Psychological Methods*, 8(4), 434-447. doi:10.1037/1082-989X.8.4.434

See Also

aov_ez and aov_car are the convenience functions to create the object appropriate for nice_anova.

Examples

```
## example from Olejnik & Algina (2003)
# "Repeated Measures Design" (pp. 439):
data(md_12.1)
# create object of class afex_aov:
rmd <- aov_ez("id", "rt", md_12.1, within = c("angle", "noise"))
rmd
nice(rmd)
str(nice(rmd))
# use different es:
nice(rmd, es = "pes") # noise: .82
nice(rmd, es = "ges") # noise: .39

# same data other approach:
rmd2 <- aov_ez("id", "rt", md_12.1, within = c("angle", "noise"),
               anova_table=list(correction = "none", es = "none"))
nice(rmd2)
nice(rmd2, correction = "GG")
nice(rmd2, correction = "GG", es = "ges")

# exampel using obk.long (see ?obk.long), a long version of the OBrienKaiser dataset from car.
data(obk.long)
# create object of class afex_aov:
tmp.aov <- aov_car(value ~ treatment * gender + Error(id/phase*hour), data = obk.long)

nice(tmp.aov, observed = "gender")

nice(tmp.aov, observed = "gender", sig_symbols = rep("", 4))

## Not run:
# use package ascii or xtable for formatting of tables ready for printing.

full <- nice(tmp.aov, observed = "gender")

require(ascii)
```

```
print(ascii(full, include.rownames = FALSE, caption = "ANOVA 1"), type = "org")

require(xtable)
print.xtable(xtable(full, caption = "ANOVA 2"), include.rownames = FALSE)

## End(Not run)
```

---

obk.long *O'Brien Kaiser's Repeated-Measures Dataset with Covariate*

Description

This is the long version of the OBrienKaiser dataset from the **car** pakage adding a random covariate age. Originally the dataset ist taken from O'Brien and Kaiser (1985). The description from OBrienKaiser says: "These contrived repeated-measures data are taken from O'Brien and Kaiser (1985). The data are from an imaginary study in which 16 female and male subjects, who are divided into three treatments, are measured at a pretest, postest, and a follow-up session; during each session, they are measured at five occasions at intervals of one hour. The design, therefore, has two between-subject and two within-subject factors."

Usage

```
obk.long
```

Format

A data frame with 240 rows and 7 variables.

Source

O'Brien, R. G., & Kaiser, M. K. (1985). MANOVA method for analyzing repeated measures designs: An extensive primer. *Psychological Bulletin*, 97, 316-333. doi:10.1037/0033-2909.97.2.316

Examples

```
# The dataset is constructed as follows:
data("OBrienKaiser", package = "carData")
set.seed(1)
OBrienKaiser2 <- within(OBrienKaiser, {
id <- factor(1:nrow(OBrienKaiser))
age <- scale(sample(18:35, nrow(OBrienKaiser), replace = TRUE), scale = FALSE)})
attributes(OBrienKaiser2$age) <- NULL # needed or resahpe2::melt throws an error.
OBrienKaiser2$age <- as.numeric(OBrienKaiser2$age)
obk.long <- reshape2::melt(OBrienKaiser2, id.vars = c("id", "treatment", "gender", "age"))
obk.long[,c("phase", "hour")] <- lapply(as.data.frame(do.call(rbind,
        strsplit(as.character(obk.long$variable), "\\."),)), factor)
obk.long <- obk.long[,c("id", "treatment", "gender", "age", "phase", "hour", "value")]
obk.long <- obk.long[order(obk.long$id),]
rownames(obk.long) <- NULL
```

```
str(obk.long)
## 'data.frame':   240 obs. of  7 variables:
##  $ id       : Factor w/ 16 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ treatment: Factor w/ 3 levels "control","A",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ gender   : Factor w/ 2 levels "F","M": 2 2 2 2 2 2 2 2 2 2 ...
##  $ age      : num  -4.75 -4.75 -4.75 -4.75 -4.75 -4.75 -4.75 -4.75 -4.75 -4.75 ...
##  $ phase    : Factor w/ 3 levels "fup","post","pre": 3 3 3 3 2 2 2 2 2 ...
##  $ hour     : Factor w/ 5 levels "1","2","3","4",..: 1 2 3 4 5 1 2 3 4 5 ...
##  $ value    : num  1 2 4 2 1 3 2 5 3 2 ...
head(obk.long)
##    id treatment gender   age phase hour value
## 1  1   control      M -4.75   pre    1     1
## 2  1   control      M -4.75   pre    2     2
## 3  1   control      M -4.75   pre    3     4
## 4  1   control      M -4.75   pre    4     2
## 5  1   control      M -4.75   pre    5     1
## 6  1   control      M -4.75  post    1     3
```

---

round_ps *Helper function which rounds p-values*

Description

p-values are rounded in a sane way: .99 - .01 to two digits, < .01 to three digits, < .001 to four digits.

Usage

```
round_ps(x)
```

Arguments

x	a numeric vector

Value

A character vector with the same length of x.

Author(s)

Henrik Singmann

Examples

```
round_ps(runif(10))

round_ps(runif(10, 0, .01))

round_ps(runif(10, 0, .001))

round_ps(0.0000000099)
```

set_sum_contrasts *Set global contrasts*

Description

These functions are simple wrappers to set contrasts globally via `options(contrasts = ...)`.

Usage

```
set_sum_contrasts()

set_deviation_contrasts()

set_effects_contrasts()

set_default_contrasts()

set_treatment_contrasts()
```

Details

`set_deviation_contrasts` and `set_effects_contrasts` are wrappers for `set_sum_contrasts`. Likewise, `set_default_contrasts` is a wrapper to `set_treatment_contrasts()`.

Value

nothing. These functions are called for their side effects to change the global options.

sk2011.1 *Data from Singmann & Klauer (2011, Experiment 1)*

Description

Singmann and Klauer (2011) were interested in whether or not conditional reasoning can be explained by a single process or whether multiple processes are necessary to explain it. To provide evidence for multiple processes we aimed to establish a double dissociation of two variables: instruction type and problem type. Instruction type was manipulated between-subjects, one group of participants received deductive instructions (i.e., to treat the premises as given and only draw necessary conclusions) and a second group of participants received probabilistic instructions (i.e., to reason as in an everyday situation; we called this "inductive instruction" in the manuscript). Problem type consisted of two different orthogonally crossed variables that were manipulated within-subjects, validity of the problem (formally valid or formally invalid) and plausibility of the problem (inferences which were consisted with the background knowledge versus problems that were inconsistent with the background knowledge). The critical comparison across the two conditions was among problems which were valid and implausible with problems that were invalid and plausible. For example, the next problem was invalid and plausible:

Usage

```
sk2011.1
```

Format

A data.frame with 640 rows and 9 variables.

Details

If a person is wet, then the person fell into a swimming pool.
A person fell into a swimming pool.
How valid is the conclusion/How likely is it that the person is wet?

For those problems we predicted that under deductive instructions responses should be lower (as the conclusion does not necessarily follow from the premises) as under probabilistic instructions. For the valid but implausible problem, an example is presented next, we predicted the opposite pattern:

If a person is wet, then the person fell into a swimming pool.
A person is wet.
How valid is the conclusion/How likely is it that the person fell into a swimming pool?

Our study also included valid and plausible and invalid and implausible problems.

Note that the factor 'plausibility' is not present in the original manuscript, there it is a results of a combination of other factors.

Source

Singmann, H., & Klauer, K. C. (2011). Deductive and inductive conditional inferences: Two modes of reasoning. Thinking & Reasoning, 17(3), 247-281. doi:10.1080/13546783.2011.572718

Examples

```
data(sk2011.1)

# Table 1 (p. 264):
aov_ez("id", "response", sk2011.1[ sk2011.1$what == "affirmation",],
       within = c("inference", "type"), between = "instruction",
       anova_table=(es = "pes"))
aov_ez("id", "response", sk2011.1[ sk2011.1$what == "denial",],
       within = c("inference", "type"), between = "instruction",
       anova_table=(es = "pes"))
```

sk2011.2 *Data from Singmann & Klauer (2011, Experiment 2)*

Description

Singmann and Klauer (2011) were interested in whether or not conditional reasoning can be explained by a single process or whether multiple processes are necessary to explain it. To provide evidence for multiple processes we aimed to establish a double dissociation of two variables: instruction type and problem type. Instruction type was manipulated between-subjects, one group of participants received deductive instructions (i.e., to treat the premises as given and only draw necessary conclusions) and a second group of participants received probabilistic instructions (i.e., to reason as in an everyday situation; we called this "inductive instruction" in the manuscript). Problem type consisted of two different orthogonally crossed variables that were manipulated within-subjects, validity of the problem (formally valid or formally invalid) and type of the problem. Problem type consistent of three levels: prological problems (i.e., problems in which background knowledge suggested to accept valid but reject invalid conclusions), neutral problems (i.e., in which background knowledge suggested to reject all problems), and counterlogical problems (i.e., problems in which background knowledge suggested to reject valid but accept invalid conclusions).

Usage

```
sk2011.2
```

Format

A data.frame with 2268 rows and 9 variables.

Details

This data set contains 63 participants in contrast to the originally reported 56 participants. The additional participants were not included in the original studies as they did not meet the inclusion criteria (i.e., no students, prior education in logic, or participated in a similar experiment). The IDs of those additional participants are: 7, 8, 9, 12, 17, 24, 30. The excluded participant reported in the paper has ID 16.

content has the following levels (C = content/conditional):
1 = Wenn eine Person in ein Schwimmbecken gefallen ist, dann ist sie nass.
2 = Wenn ein Hund Flöhe hat, dann kratzt er sich hin und wieder.
3 = Wenn eine Seifenblase mit einer Nadel gestochen wurde, dann platzt sie.
4 = Wenn ein Mädchen Geschlechtsverkehr vollzogen hat, dann ist es schwanger.
5 = Wenn eine Pflanze ausreichend gegossen wird, dann bleibt sie grün.
6 = Wenn sich eine Person die Zähne putzt, dann bekommt sie KEIN Karies.
7 = Wenn eine Person viel Cola trinkt, dann nimmt sie an Gewicht zu.
8 = Wenn eine Person die Klimaanlage angeschaltet hat, dann fröstelt sie.
9 = Wenn eine Person viel lernt, dann wird sie in der Klausur eine gute Note erhalten.

Source

Singmann, H., & Klauer, K. C. (2011). Deductive and inductive conditional inferences: Two modes of reasoning. Thinking & Reasoning, 17(3), 247-281. doi:10.1080/13546783.2011.572718

Examples

```
data("sk2011.2")

## remove excluded participants:

sk2_final <- droplevels(sk2011.2[!(sk2011.2$id %in% c(7, 8, 9, 12, 16, 17, 24, 30)),])
str(sk2_final)

## Table 2 (inference = problem):
aov_ez("id", "response", sk2_final[sk2_final$what == "affirmation",],
       between = "instruction", within = c("inference", "type"),
       anova_table=list(es = "pes"))

aov_ez("id", "response", sk2_final[sk2_final$what == "denial",],
       between = "instruction", within = c("inference", "type"),
       anova_table=list(es = "pes"))

# Recreate Figure 4 (corrected version):

sk2_aff <- droplevels(sk2_final[sk2_final$what == "affirmation",])
sk2_aff$type2 <- factor(sk2_aff$inference:sk2_aff$type, levels = c("MP:prological",
                            "MP:neutral", "MP:counterlogical", "AC:counterlogical",
                            "AC:neutral", "AC:prological"))
a1_b <- aov_ez("id", "response", sk2_aff,
       between = "instruction", within = c("type2"))

sk2_den <- droplevels(sk2_final[sk2_final$what == "denial",])
sk2_den$type2 <- factor(sk2_den$inference:sk2_den$type, levels = c("MT:prological",
                            "MT:neutral", "MT:counterlogical", "DA:counterlogical",
                            "DA:neutral","DA:prological"))
a2_b <- aov_ez("id", "response", sk2_den,
       between = "instruction", within = c("type2"))

if (requireNamespace("emmeans")) {
  emmeans::emmip(a1_b,instruction~type2, ylim = c(0, 100))
  emmeans::emmip(a2_b,instruction~type2, ylim = c(0, 100))
}
```

`test_levene` *Assumption Tests for ANOVAs*

Description

`test_levene` computes Levene's test for homogeneity of variances across groups via `car::leveneTest`. `test_sphericity` computes Mauchly test of sphericity via `car::Anova`.

Usage

```
test_levene(afex_aov, center = mean, ...)

test_sphericity(afex_aov)
```

Arguments

afex_aov	afex_aov object.
center	Function to compute the center of each group; mean (the default) gives the original Levene's test.
...	passed to leveneTest

Author(s)

Mattan S. Ben-Shachar and Henrik Singmann

Examples

```
### Setup ANOVAs
data(obk.long, package = "afex")
between_1 <- aov_car(value ~ treatment + Error(id), data = obk.long)
between_2 <- aov_car(value ~ treatment*gender + Error(id), data = obk.long)
mixed <- aov_car(value ~ treatment * gender + Error(id/(phase*hour)), data = obk.long)
within <- aov_car(value ~ 1 + Error(id/(phase*hour)), data = obk.long)

### Levene Test for Homogeneity of Variances
test_levene(between_1)
test_levene(between_2)
test_levene(mixed)
## Not run:
test_levene(within) ## fails

## End(Not run)

### Mauchly Test of Sphericity
## Not run:
## fails for between-subjects only models:
test_sphericity(between_1)
test_sphericity(between_2)

## End(Not run)
test_sphericity(mixed)
test_sphericity(within)
```

Index