Package ‘afex’

February 19, 2015

Type Package

Title Analysis of Factorial Experiments

Depends R (>= 3.0.0), car, lme4 (>= 1.0.5), reshape2

Suggests ascii, xtable, parallel, plyr, optimx, nloptr

Imports stringr, coin, Matrix, pbkrtest (>= 0.3-6)

Description Provides convenience functions for analyzing factorial experiments using ANOVA or mixed models. ez.glm(), aov.car(), and aov4() allow convenient specification of between, within (i.e., repeated-measures), or mixed between-within (i.e., split-plot) ANOVAs for data in long format (i.e., one observation per row), potentially aggregating multiple observations per individual and cell of the design. mixed() fits a mixed model using lme4::lmer() and computes p-values for all effects using either Kenward-Roger approximation for degrees of freedom (LMM only), parametric bootstrap (LMMs and GLMMs), or likelihood ratio tests (LMMs and GLMMs). afex uses type 3 sums of squares as default (imitating commercial statistical software). compare.2.vectors() compares two vectors using a variety of tests (t, Wilcoxon, and permutation).

URL http://www.psychologie.uni-freiburg.de/Members/singmann/R/afex

License GPL (>= 3)

Encoding latin1

Version 0.13-145

Date 2015-01-09

Author Henrik Singmann [aut, cre],
    Ben Bolker [aut],
    Jake Westfall [aut],
    Søren Højsgaard [ctb],
    John Fox [ctb],
    Michael A. Lawrence [ctb],
    Ulf Mertens [ctb]

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

Repository CRAN

Repository/R-Forge/Project afe
The afew Package

Description

Analysis of Factorial Experiments.

Details

Package: afex
Type: Package
Version: 0.13-145
Date: 2015-01-09
Depends: R (>= 3.0.0), car, lme4 (>= 1.0.5), reshape2
Encoding: latin1
License: GPL (>=3)
URL: http://www.psychologie.uni-freiburg.de/Members/singmann/R/afex

Provides convenience functions for analyzing factorial experiments using ANOVA or mixed models. ez.glm(), aov.car(), and aov4() allow convenient specification of between, within (i.e., repeated-
allFit measures), or mixed between-within (i.e., split-plot) ANOVAs for data in long format (i.e., one observation per row), potentially aggregating multiple observations per individual and cell of the design. mixed() fits a mixed model using lme4::lmer() and computes p-values for all effects using either Kenward-Roger approximation for degrees of freedom (LMM only), parametric bootstrap (LMMs and GLMMs), or likelihood ratio tests (LMMs and GLMMs). afex uses type 3 sums of squares as default (imitating commercial statistical software). compare.2.vectors() compares two vectors using a variety of tests (t, Wilcoxon, and permutation).

Author(s)
Henrik Singmann, Ben Bolker, Jake Westfall, with contributions from Søren Højsgaard, John Fox, Michael A. Lawrence, Ulf Mertens

---

**allFit**

*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 three 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.

**Usage**

```r
allFit(m, meth.tab = cbind(optimizer = rep(c("bobyqa", "Nelder_Mead", "optimx", "nloptwrap"), c(1, 1, 2, 2)), method = c("", ",", "nlminb", "L-BFGS-B", "NLOPT_LN_NELDERMEAD", "NLOPT_LN_BOBYQA")), verbose = TRUE, maxfun = 1e+05, ...
```

**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 routine.
- `...` further arguments passed to `update.merMod` such as data.

**Details**

Needs packages `nloptr` and `optimx` to try out all optimizers. `optimx` needs to be loaded explicitly using `library` or `require`. 
Value

a list of fitted merMod objects

Author(s)

Ben Bolker

See Also

slice, slice2D in the bbmle package

Examples

```r
## Not run:
require(optimx)
gm1 <- glmer(cbind(incidence, size - incidence) ~ period + (1 | herd),
data = cbpp, family = binomial)
gm_all <- allFit(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?

## End(Not run)
```

---

**aov.car**

*Convenience wrappers for car::Anova using either a formula or factor based interface.*

Description

These functions allow convenient access to **Anova** (from the **car** package) for data in the **long** format (i.e., one observation per row), possibly aggregating the data if there is more than one observation per individual and cell. Hence, mixed between-within ANOVAs can be calculated conveniently without using the rather unhandy format of **car::Anova**. **aov.car** can be called using a formula similar to **aov** specifying an error strata for the within-subject factor(s), **aov4** can be called with a **lme4**-like formula, and **ez.glm** is called specifying the factors as character vectors.

Usage

```r
aov.car(formula, data, fun.aggregate = NULL, type = 3,
  factorize = TRUE, check.contrasts = TRUE,
  return = "nice", observed = NULL, args.return = list(), ...)
```

```r
aov4(formula, data, observed = NULL, fun.aggregate = NULL, type = 3,
  factorize = TRUE, check.contrasts = TRUE,
  return = "nice", args.return = list(), ...,
  print.formula = FALSE)
```
Arguments

formula A formula specifying the ANOVA model similar to `aov` (for `aov.car` or similar to `lme4::lmer` for `aov`). Should include an error term (i.e., `Error(id/)` or `(. . . |id)`). 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`.

type The type of sums of squares for the ANOVA. **Defaults 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. Default is `TRUE`. If one wants to run an ANCOVA, 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.

return What should be returned? If "nice" (the default) will return a nice ANOVA table (produced by `nice.anova`). Possible values are c("Anova", "lm", "data", "nice", "full", "all", possibly abbreviated).

observed character vector indicating which of the variables are observed (i.e, measured) as compared to experimentally manipulated. The default behavior is to return an ANOVA table with generalized eta squared effect size for which this information is necessary (see `nice.anova`).

args.return list of further arguments passed to the function which produces the return value. Currently only supports `return = "nice"` (the default) which then passes arguments to `nice.anova` (see examples).

... Further arguments passed to `fun.aggregate`.

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) 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.

print.formula: ez.glm is a wrapper for aov.car. This boolean argument indicates whether the formula in the call to car.aov should be printed.

object: An object of class Anova.mlm as returned by aov.car, ez.glm, or Anova.

Details

Type 3 sums of squares are default in afex. Note that type 3 sums of squares are said to be dangerous and/or problematic. On the other side they are the default in in SPSS and SAS and recommended by e.g. Maxwell and Delaney (2004). For a brief discussion see here.

However, 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 \texttt{contr.sum} via options(contrasts=c('contr.sum', 'contr.poly')). This should be done automatically when loading afex and afex will issue a warning when running type 3 SS and other coding schemes. You can check the coding with options("contrasts").

The formulas for aov.car or aov4 must contain a single Error term specifying the ID column and potential within-subject factors (you can use mixed 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, i.e., it is not necessary to specify them outside the Error term, see Examples).

Suppressing the intercept (i.e, via \texttt{\texttt{~} + 0} or \texttt{- 1}) is ignored. Specific specifications of effects (e.g., excluding terms with \texttt{\texttt{~} -} or using \texttt{\texttt{\texttt{\*}}} could be okay but is not tested. Using the \texttt{I} or \texttt{poly} function within the formula is not tested and not supported!

For ez.glm either between or within must not be NULL.

ez.glm will concatenate all between-subject factors using \texttt{\*} (i.e., producing all main effects and interactions) and all covariates by \texttt{+} (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.

To run an ANCOVA you need to set factorize = \texttt{\texttt{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 return a \texttt{nice.anova} data.frame. This includes calculation of generalized eta squared for which all non manipulated (i.e., observed) variables need to be specified via the observed argument. Changing the effect size to "pes" (partial eta-squared) via args.return or the return value via return removes this necessity.

If check.contrasts = \texttt{\texttt{TRUE}}, contrasts will be set to "contr.sum" for all between-subject factors if default contrasts are not equal to "contr.sum" or attr(factor, "contrasts") != "contr.sum". (within-subject factors are hard-coded "contr.sum").

Value

aov.car, aov4, and ez.glm are wrappers to \texttt{Anova}, the return value is dependent on the return argument. When argument return is "nice" (the default) a nice ANOVA table is returned (\texttt{nice.anova}) with the following columns: Effect, df, MSE (mean-squared errors), F (potentially with significant symbols), ges (generalized eta-squared), p.

If return = "full" or return = "all" a list list with the following elements:
"Anova" the same as Anova. Usually an object of class "Anova.mlm" (with within-subjects factors) or of class c("anova", "data.frame"). Also returned if return = "Anova".

"lm" the object fitted with lm and passed to Anova (i.e., an object of class "lm" or "m1m"). Also returned if return = "lm".

"data" the data used to fit the lm object. Also returned if return = "data".

"idata" if within-subject factors are present, the idata argument passed to Anova.

"marginal" a list containing the marginal means (the same as when return = "marginal").

If return = "univariate" the object returned from univ.

univ returns a list of data.frames containing the univariate results (i.e., the classical ANOVA results) from an object of class "Anova.mlm". This is essentially the output from summary.Anova.mlm with multivariate = FALSE, e.g. summary(aov.car(...), multivariate = FALSE), as a list instead of printed to the console.

For objects of class "anova" (i.e., the object returned by car::Anova for a purely between-subjects ANOVA) the object is returned unaltered.

The elements of the list returned by univ are: anova, mauchly, and spehricity.correction (containing both, Greenhouse-Geisser and Hyundt-Feldt correction).

If return = "marginal" A list with data.frames containing the marginal means for all effects. Numerical variables are ignored.

If return = "aov", an object returned from aov with the (possibly aggregated) data fitted with aov using "contr.sum" for all factors as long as check.contrasts = TRUE. This object can be passed to lsmeans::lsmeans for post-hoc tests or lsmeans::lsmip for plotting. Note that aov is not reliably working for unbalanced data.

**Note**

The id variable and variables entered as within-subjects (i.e., repeated-measures) factors are silently converted to factors. 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. aov4 and ez.glm 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 aov).

**Author(s)**

univ is basically a copy of summary.Anova.mlm written by John Fox.

The other functions were written by Henrik Singmann.

The design of these functions is heavily influenced by ezANOVA from package ez.

**References**

See Also

nice.anova creates the nice ANOVA tables which are by default returned. 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.

Examples

# Examples from a purely within-design from
# Maxwell & Delaney (2004, Chapter 11),
# Table 12.5 (p. 578):
data(md_12.1)
ez.glm("id", "rt", md_12.1, within = c("angle", "noise"),
     args.return=list(correction = "none", es = "none"))

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

data(obk.long, package = "afex")

# run univariate mixed ANOVA for the full design:
aov.car(value ~ treatment * gender + Error(id/(phase*hour)),
     data = obk.long, observed = "gender")
aov4(value ~ treatment * gender + (phase*hour|id),
     data = obk.long, observed = "gender")
ez.glm("id", "value", obk.long, between = c("treatment", "gender"),
     within = c("phase", "hour"), observed = "gender")

# both calls return the same:
##                  Effect df MSE F  ges  p
## 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
# replicating ?Anova using aov.car:
aov.car(value ~ treatment * gender + Error(id/(phase*hour)),
data = obk.long, type = 2, return = "Anova")
# in contrast to aov you do not need the within-subject factors outside Error()

# replicating ?Anova using ez.glm:
ez.glm("id", "value", obk.long, c("treatment", "gender"),
c("phase", "hour"), type = 2, return = "Anova")

#both return:
## 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 | 2 | 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

# 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)
aov4(value ~ treatment * gender + age + (phase*hour|id),
data = obk.long, observed = c("gender", "age"), factorize = FALSE)

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

# runs with "numeric" factors
obk.long$hour2 <- as.numeric(as.character(obk.long$hour))
aov.car(value ~ treatment * gender + Error(id/hour2),
data = obk.long, type = 2, observed = c("gender"))
# only between
aov.car(value ~ treatment * gender + Error(id),
data = obk.long, observed = c("gender"))
aov4(value ~ treatment * gender + (1|id), print.formula = TRUE,
data = obk.long, observed = c("gender"))
ez.glm("id", "value", obk.long, c("treatment", "gender"),
       within = NULL, print.formula = TRUE, observed = "gender")

# only within
aov.car(value ~ Error(id/(phase*hour)), data = obk.long, type = 2)
aov4(value ~ (phase*hour|id), data = obk.long, type = 2,
     print.formula = TRUE)
ez.glm("id", "value", obk.long, NULL, c("phase", "hour"),
       type = 2, print.formula = TRUE)

# using return = "full":
str(aov.car(value ~ Error(id/(phase*hour)), data = obk.long, return = "full"), 1)

## List of 4
## $ Anova:List of 14
## ... attr(*, "class")= chr "Anova.mlm"
## $ lm :List of 11
## ... attr(*, "class")= chr [1:2] "lm" "lm"
## $ data :'data.frame': 16 obs. of 16 variables:
## $ idata:'data.frame': 15 obs. of 2 variables:
## $ marginal:List of 3

# use args.return arguments:
 aov.car(value ~ treatment * gender + Error(id/(phase*hour)),
           data = obk.long, args.return = list(correction = "none", es = "pes"))
aov.car(value ~ treatment * gender + Error(id/(phase*hour)),
           data = obk.long, observed = "gender",
           args.return = list(correction = "none", MSE = FALSE))

---

**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

```r
compare.2.vectors(x, y, paired = FALSE, na.rm = FALSE,
                  tests = c("parametric", "nonparametric"), coin = TRUE,
                  alternative = "two.sided",
                  perm.distribution = approximate(100000),
                  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 `Distribution` or `IndependenceTest`. Defaults to approximate(100000) indicating an approximation of the exact 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/Slatterwaith/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 selection 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., parametric and/or nonparametric) each containing a data.frame with the following columns: test, test.statistic, test.value, test.df, p.

Examples

```r
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))
```

---

**Description**

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

**Usage**

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

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](http://stats.stackexchange.com/a/122662/442).

Examples

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

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

# EMSs for Clark design if Words are fixed
esms(r ~ S*W*T, nested="T/W", random="S")
```
Data 12.1 from Maxwell & Delaney

Description

Hypothetical Reaction Time Data for 2 x 3 Perceptual Experiment: Example data for chapter 12 of Maxwell 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

md_15.1

Examples

data(md_12.1)

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

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

Examples

### Replicate results from Table 15.2 to 15.6 (Maxwell & Delaney, 2004, pp. 774)
```r
data(md_16.1)
```

### ANOVA results (Table 15.2)
```r
aov4(iq ~ time + (timecat|id), data=md_15.1, args.return=list(correction = "none"))
```

### Table 15.3 (random intercept only)
```r
# 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)
```

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

### Table 15.4, page 789
```r
# 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
```r
# set up polynomial contrasts for timecat
contrasts(md_15.1$angle) <- contr.poly
# fit all parameters separately
(t15.5 <- mixed(iq ~ timecat + (1+time|id), data=md_15.1, check.contrasts=FALSE, 
              per.parameter="angle")
# quadratic trend is considerably off, conclusions stay the same.
```

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

---

### Description

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

### Usage
```
md_16.1
```
Data from a hypothetical inductive reasoning study.
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


Examples

```r
# 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.
```

Description

Fits and calculates p-values for all effects in a mixed model fitted with `lmer()`. The default behavior calculates type 3 like p-values using the Kenward-Roger approximation for degrees-of-freedom implemented in `KRmodcomp` (for LMMs only), but also allows for parametric bootstrap
(method = "PB"), or likelihood ratio tests (the latter two for LMMs and GLMMs). `print`, `summary`, and `anova` methods for the returned object of class "mixed" are available (the last two return the same `data.frame`).

**Usage**

```r
mixed(formula, data, type = 3, method = c("KR", "PB", "LRT"),
       per.parameter = NULL, args.test = list(), test.intercept = FALSE,
       check.contrasts = TRUE, set.data.arg = TRUE, progress = TRUE,
       cl = NULL, ...)
```

**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. Only type 3 tests (3 or "III") are correctly implemented (see Details).
- `method`: character vector indicating which methods for obtaining p-values should be used. "KR" (the default) corresponds to the Kenward-Roger approximation for degrees of freedom (only working with linear mixed models). "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).
- `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. Relatively untested so results should be compared with a second run without setting this argument. 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.
- `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? Otherwise the name will be `data`. Helpful if fitted objects are used afterwards (e.g., using `lsmeans`). Default is `TRUE`.
- `progress`: if `TRUE`, shows progress with a text progress bar and other status messages during fitting.
- `cl`: A vector identifying a cluster; used for distributing the estimation of the different models using several cores. See examples. If `check.contrasts`, 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.
... further arguments (such as weights) passed to lmer.

Details

For an introduction to mixed-modeling for experimental designs see Barr, Levy, Scheepers, & Tily (2013; I highly recommend reading this paper if you use this function), 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).

p-values are per default calculated via methods from pbkrtest. When method = "KR" (the default), the Kenward-Roger approximation for degrees-of-freedom is calculated using KRmodcomp, which is only applicable to linear-mixed models. The test statistic in the output is a F-value (f).

method = "PB" calculates p-values using parametric bootstrap using PBmodcomp. This can be used for linear and also generalized linear mixed models (GLMM) by specifying a family argument to mixed. Note that you should specify further arguments to PBmodcomp via argsNtest, 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 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.

Type 3 tests are obtained by comparing a model in which only the tested effect is excluded with the full model (containing all effects). This corresponds to the (type 3) Wald tests given by car::Anova for "merMod" models. The submodels in which the tested effect is excluded are obtained by manually creating a model matrix which is then fitted in "lme4". This is done to avoid R's "feature" to not allow this behavior.

Type 2 tests are truly sequential. They are obtained by comparing a model in which the tested effect and all higher order 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 you want type 2 Wald tests instead of truly sequential type 2 tests, use car::Anova with test = "F". Note that the order in which the effects are entered into the formula does not matter (in contrast to type 1 tests).

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 attr(factor, "contrasts") != "contr.sum". Furthermore, the current contrasts (obtained via getOption("contrasts")) will be set at the cluster nodes if cl is not NULL.
mixed

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 \texttt{KRmodcomp}. The stat column in this data.frame gives the value of the test statistic, an F-value for \texttt{method = "KR"} and a chi-square value for the other two methods.
2. full.model the "\texttt{LmerMod}" object returned from fitting the full mixed model.
3. restricted.models a list of "\texttt{LmerMod}" objects from fitting 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. type The \texttt{type} argument used when calling this function.
6. method The \texttt{method} argument used when calling this function.

Three identical methods exist for objects of class "mixed": \texttt{print}, \texttt{summary}, and \texttt{anova}. They all 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. \texttt{p.value} estimated p-value for the effect

For LMMs with \texttt{method="KR"} 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 \texttt{http://stackoverflow.com/a/25612960/289572}):

1. \texttt{f} computed F statistic
2. \texttt{ndf} numerator degrees of freedom (number of parameters used for the effect)
3. \texttt{ddf} denominator degrees of freedom (effective residual degrees of freedom for testing the effect), computed from the Kenward-Roger correction using \texttt{pbkrtest::KRmodcomp}
4. \texttt{f.scaling} scaling of F-statistic computing from Kenward-Roger approximation.

For models with \texttt{method="LRT"} the following further columns are returned:

1. \texttt{df.large} degrees of freedom (i.e., estimated parameters) for full model (i.e., model containing the corresponding effect)
2. \texttt{df.small} degrees of freedom (i.e., estimated parameters) for restricted model (i.e., model without the corresponding effect)
3. \texttt{chisq} 2 times the difference in likelihood (obtained with \texttt{logLik}) between full and restricted model
4. \texttt{df} difference in degrees of freedom between full and restricted model (p-value is based on these df).

For models with \texttt{method="PB"} the following further column is returned:

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

When method = "KR", obtaining p-values is known to crash due to 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 "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).

Please report all bugs to henrik.singmann (at) psychologie.uni-freiburg.de

Author(s)

Henrik Singmann with contributions from Ben Bolker and Joshua Wiley.

References


See Also

ez.glm and aov.car for convenience functions to analyze experimental designs 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

```r
### replicate results from Table 15.4 (Maxwell & Delaney, 2004, p. 789)
data(md_15.1)
# random intercept plus slope
(t15.4a <- mixed(lq ~ 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)$timecat <- contr.treatment(4, base = 4)
(t15.4b <- mixed(lq ~ timecat + (1+time|id), data=md_15.1, check.contrasts=FALSE))
summary(t15.4a$full.model) # gives "wrong" parameters estimates
summary(t15.4b$full.model) # 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 due to the use of Kenward-Roger approximation here,
# whereas MNW's p-values are based on uncorrected df.)

# again, to obtain identical parameter and t-values, use treatment contrasts:
summary(mixed2$full.model) # 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$full.model)

### 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$full.model)

## 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:
data(obk.long, package = "afex")

obk.long$hour <- ordered(obk.long$hour)

# tests only the main effect parameters of hour individually per parameter.
mixed(value ~ treatment*phase*hour + (1|id), per.parameter = "hour", data = obk.long)

# tests all parameters including hour individually
mixed(value ~ treatment*phase*hour + (1|id), per.parameter = "hour", data = obk.long)

# tests all parameters individually
mixed(value ~ treatment*phase*hour + (1|id), per.parameter = ".", data = obk.long)

# 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

##   Effect stat ndf  ddf F.scaling p.value
## 1 Correct   8.15  1  1627.73     1.00  .004
# Fitting a GLMM using parametric bootstrap:
require("mlmRev")  # for the data, see _contraception

gml <- mixed(use ~ age + I(age^2) + urban + livch + (1 | district), method = "PB",
family = binomial, data = Contraception, args.test = list(nsim = 10))

# Using multicore

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

## There are two ways to use multicore:

# 1. Obtain fits with multicore:
mixed(value ~ treatment*phase*hour +(1|id), data = obk.long, method = "LRT", cl = cl)

# 2. Obtain PB samples via multicore:
mixed(use ~ age + I(age^2) + urban + livch + (1 | district), family = binomial,
method = "PB", data = Contraception, args.test = list(nsim = 10, cl = cl))

## Both ways can be combined:
mixed(use ~ age + I(age^2) + urban + livch + (1 | district), family = binomial,
method = "PB", data = Contraception, args.test = list(nsim = 10, cl = cl), cl = cl)

stopCluster(cl)

## End(Not run)

---

### nice.anova

Make nice ANOVA table for printing.

#### Description

These functions produce a nice ANOVA table best for printing. `nice.anova` takes an object from `Anova` possible created by the convenience functions `ez.glm` or `aov.car`. When within-subject factors are present, either sphericity corrected or uncorrected degrees of freedom can be reported.
Usage

nice.anova(object, es = "ges", observed = NULL, correction = c("GG", "HF", "none"),
MSE = TRUE, intercept = FALSE, sig.symbols = c(" +", " *", " **", " ***"))

Arguments

object  
An object of class "Anova.mlm" or "anova" as returned from Anova, ez.glm,
or aov.car.

es  
Effect Size to be reported. Default is "ges", which reports 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 on the degrees of freedom should be
reported for the within-subject factors. The default c("GG", "HF", "none")
corresponds to the Greenhouse-Geisser correction.

MSE  
logical. Should the column containing the Mean Squared Error (MSE) be dis-
played? Default is TRUE.

intercept  
logical. Should intercept (if present) be printed (default is FALSE which sup-
presses printing of the intercept)

sig.symbols  
Character. What should be the symbols designating significance? When enter-
ing 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.

Details

The returned data.frame is print-ready when adding to a document with proper methods. I recom-
med ascii and xtable. ascii provides conversion to AsciiDoc but most notably to org-mode (see
ascii and print.ascii). xtable converts a data.frame into LaTeX code with many possible op-
tions (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 Repro-
ducible 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.

Value

A data.frame 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


See Also

ez.glm and aov.car are the convenience functions to create the object appropriate for nice.anova.

Examples

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

# example using obk.long (see ?obk.long), a long version of the OBrienKaiser dataset from car.
data(obk.long)
# create object of class Anova:
tmp.aov <- aov.car(value ~ treatment * gender + Error(id/phase*hour),
                   data = obk.long, return = "Anova")
nice.anova(tmp.aov, observed = "gender")

nice.anova(tmp.aov, observed = "gender", sig.symbol = rep("", 4))

## Not run:
# use package ascii or xtable for formatting of tables ready for printing.
full <- nice.anova(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)
```
Description

This is the long version of the O'BrienKaiser dataset from the car package adding a random covariate age. Originally the dataset is taken from O'Brien and Kaiser (1985). The description from O'BrienKaiser 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


Examples

# The dataset is constructed as follows:
set.seed(1)
O'BrienKaiser2 <- within(O'BrienKaiser, {
id <- factor(1:nrow(O'BrienKaiser))
age <- scale(sample(18:35, nrow(O'BrienKaiser), replace = TRUE), scale = FALSE))
attributes(O'BrienKaiser2$age) <- NULL # needed or resahpe::melt throws an error.
O'BrienKaiser2$age <- as.numeric(O'BrienKaiser2$age)
obk.long <- reshape2::melt(O'BrienKaiser2, id.vars = c("id", "treatment", "gender", "age"))
obk.long[,c("phase", "hour")][ < - lapply(as.data.frame(do.call(rbind,
strsplit(as.character(obk.long$variable), "\.")",)), factor)
obk.long <- obk.long[,c("id", "treatment", "gender", "age", "phase", "hour", "value")]
rownames(obk.long) <- order(obk.long$id,]
str(obk.long)
## 'data.frame': 240 obs. of 7 variables:
## $ id : Factor w/ 16 levels "1","2","3","4",...: 11111111111111
## $ treatment: Factor w/ 3 levels "control","A",...: 11111111111111
## $ age : Factor w/ 2 levels "F","M": 22222222222222
## $ phase : Factor w/ 3 levels "fup","post","pre": 3333332222222222
## $ hour : Factor w/ 5 levels "1","2","3","4",...: 1234512345...
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

```r
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.
# Index

**Topic dataset**
- md_12.1, 14
- md_15.1, 15
- md_16.1, 16
- md_16.4, 17
- obk.long, 28

**Topic package**
- afex-package, 2
- allFit, 3
- Anova, 4–7, 22, 25, 26
- aov, 4, 5, 7
- aov.car, 4, 22, 25–27
- aov4 (aov.car), 4
- ascii, 26
- compare.2.vectors, 10
- contr.sum, 6
- Distribution, 11
- ems, 12
- ez.glm, 22, 25–27
- ez.glm (aov.car), 4
- ezANOVA, 7
- family, 20
- grep, 19
- I, 6
- IndependenceTest, 11
- KRmodcomp, 18, 20, 21
- lmer, 18, 20
- md_12.1, 14
- md_15.1, 15, 22
- md_16.1, 16, 22
- md_16.4, 17, 22
- mean, 5
- mixed, 6, 8, 18
- nice.anova, 5, 6, 8, 25
- obk.long, 28
- OBrienKaiser, 28
- oneway.test, 11
- PBmodcomp, 20
- poly, 6
- print.xtable, 26
- regex, 19
- round_ps, 29
- set.default_contrasts (set_sum_contrasts), 30
- set.deviation_contrasts (set_sum_contrasts), 30
- set.effects_contrasts (set_sum_contrasts), 30
- set.sum_contrasts, 30
- set.treatment_contrasts (set_sum_contrasts), 30
- summary.Anova.mlm, 7
- t.test, 11
- univ (aov.car), 4
- update.merMod, 3
- wilcox.test, 11
- wilcox_test, 11
- xtable, 26