Package: ez (via r-universe)

May 19, 2026

Version 4.5-0

Date 2026-02-07

Title Easy Analysis and Visualization of Factorial Experiments

URL https://github.com/bucky2177/ez

Depends R (>= 3.1)

Imports car (>= 2.1-3), ggplot2 (>= 3.0.0), lme4 (>= 1.1-12), MASS (>= 7.3-45), Matrix (>= 1.2-7.1), mgcv (>= 1.8-12), plyr (>= 1.8.4), reshape2 (>= 1.4.2), scales (>= 0.4.0), stringr (>= 1.1.0), rlang (>= 0.4.0)

Description Facilitates easy analysis of factorial experiments, including purely within-Ss designs (a.k.a. ``repeated measures''), purely between-Ss designs, and mixed within-and-between-Ss designs. The functions in this package aim to provide simple, intuitive and consistent specification of data analysis and visualization. Visualization functions also include design visualization for pre-analysis data auditing, and correlation matrix visualization. Finally, this package includes functions for non-parametric analysis, including permutation tests and bootstrap resampling. The bootstrap function obtains predictions either by cell means or by more advanced/powerful mixed effects models, yielding predictions and confidence intervals that may be easily visualized at any level of the experiment's design.

License GPL (>= 2)

LazyLoad yes

ByteCompile true

Encoding UTF-8

RoxygenNote 7.3.3

Config/pak/sysreqs cmake make libicu-dev

Repository https://bucky2177.r-universe.dev

Date/Publication 2026-02-16 09:58:07 UTC

RemoteUrl https://github.com/bucky2177/ez

RemoteRef HEAD

RemoteSha c2f67f92289f602f505f4f9df929d3aba44e80d5

Contents

`ez-package` *Easy analysis and visualization of factorial experiments*

Description

This package facilitates easy analysis of factorial experiments, including purely within-Ss designs (a.k.a. "repeated measures"), purely between-Ss designs, and mixed within-and-between-Ss designs. The functions in this package aim to provide simple, intuitive and consistent specification of data analysis and visualization. Visualization functions also include design visualization for pre-analysis data auditing, and correlation matrix visualization. Finally, this package includes functions for non-parametric analysis, including permutation tests and bootstrap resampling. The bootstrap function obtains predictions either by cell means or by more advanced/powerful mixed effects models, yielding predictions and confidence intervals that may be easily visualized at any level of the experiment's design.

Details

Package:	ez
Type:	Package
Version:	4.5-0
Date:	2016-11-01

License: GPL-3
LazyLoad: yes

This package contains several useful functions:

`ezANOVA` Provides simple interface to ANOVA, including assumption checks.

`ezBoot` Computes bootstrap resampled cell means or lmer predictions

`ezCor` Function to plot a correlation matrix with scatterplots, linear fits, and univariate density plots

`ezDesign` Function to plot a visual representation of the balance of data given a specified experimental design. Useful for diagnosing missing data issues.

`ezMixed` Provides assessment of fixed effects in a mixed effects modelling context.

`ezPerm` Provides simple interface to the Permutation test.

`ezPlot` Uses the `ggplot2` graphing package to generate plots for any given user-requested effect, by default producing error bars that facilitate visual post-hoc multiple comparisons.

`ezPlot2` When supplied the results from a call to `ezPredict` or `ezBoot`, plots predictions with confidence intervals.

`ezPrecis` Provides a summary of a given data frame.

`ezPredict` Computes predicted values from the fixed effects of a mixed effects model.

`ezResample` Resamples data, useful when bootstrapping.

`ezStats` Provides between-Ss descriptive statistics for any given user-requested effect.

This package also contains two data sets:

`ANT` Simulated data from the Attention Network Test

`ANT2` Messy version of the ANT data set

Author(s)

Michael A. Lawrence <mike.lwrnc@gmail.com>
Visit the ez development site at https://github.com/mike-lawrence/ez
for the bug/issue tracker and the link to the mailing list.

See Also

`ANT`, `ANT2`, `ezANOVA`, `ezBoot`, `ezCor`, `ezDesign`, `ezMixed`, `ezPerm`, `ezPlot`, `ezPlot2`, `ezPrecis`, `ezPredict`, `ezResample`, `ezStats`

ANT *ANT data*

Description

Simulated data from then Attention Network Test (see reference below), consisting of 2 within-Ss variables ("cue" and "flank"), 1 between-Ss variable ("group") and 2 dependent variables (response time, "rt", and whether an error was made, "error")

Usage

```
data(ANT)
```

Format

A data frame with 5760 observations on the following 10 variables.

`subnum` a factor with levels `1` `2` `3` `4` `5` `6` `7` `8` `9` `10` `11` `12` `13` `14` `15` `16` `17` `18` `19` `20`

`group` a factor with levels `Control` `Treatment`

`block` a numeric vector

`trial` a numeric vector

`cue` a factor with levels `None` `Center` `Double` `Spatial`

`flank` a factor with levels `Neutral` `Congruent` `Incongruent`

`location` a factor with levels `down` `up`

`direction` a factor with levels `left` `right`

`rt` a numeric vector

`error` a numeric vector

Author(s)

Michael A. Lawrence <mike.lwrnc@gmail.com>
Visit the ez development site at https://github.com/mike-lawrence/ez
for the bug/issue tracker and the link to the mailing list.

References

J Fan, BD McCandliss, T Sommer, A Raz, MI Posner (2002). Testing the efficiency and independence of attentional networks. *Journal of Cognitive Neuroscience*, **14**, 340-347.

Examples

```
data(ANT)
head(ANT)
ezPrecis(ANT)
```

ANT2 *Messy ANT data*

Description

A "messy" version of the ANT data set (see ANT). In this version of the data, subnum #7 is missing data from the last half of the experiment, subnum #14 made all errors in the incongruent cells, and subnum #12 mistakenly reversed their responses.

Usage

```
data(ANT2)
```

Author(s)

Michael A. Lawrence <mike.lwrnc@gmail.com>
Visit the ez development site at https://github.com/mike-lawrence/ez
for the bug/issue tracker and the link to the mailing list.

Examples

```
data(ANT2)
head(ANT2)
ezPrecis(ANT2)
```

ezANOVA *Compute ANOVA*

Description

This function provides easy analysis of data from factorial experiments, including purely within-Ss designs (a.k.a. "repeated measures"), purely between-Ss designs, and mixed within-and-between-Ss designs, yielding ANOVA results, generalized effect sizes and assumption checks.

Usage

```
ezANOVA(
    data
    , dv
    , wid
    , within = NULL
    , within_full = NULL
    , within_covariates = NULL
    , between = NULL
    , between_covariates = NULL
    , observed = NULL
```

```
    , diff = NULL
    , reverse_diff = FALSE
    , type = 2
    , white.adjust = FALSE
    , detailed = FALSE
    , return_aov = FALSE
)
```

Arguments

data	Data frame containing the data to be analyzed.
dv	Name of the column in data that contains the dependent variable. Values in this column must be numeric.
wid	Name of the column in data that contains the variable specifying the case/Ss identifier. This should be a unique value per case/Ss.
within	Names of columns in data that contain predictor variables that are manipulated (or observed) within-Ss. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.
within_full	Same as within, but intended to specify the full within-Ss design in cases where the data have not already been collapsed to means per condition specified by within and when within only specifies a subset of the full design.
within_covariates	Names of columns in data that contain predictor variables that are manipulated (or observed) within-Ss and are to serve as covariates in the analysis. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.
between	Names of columns in data that contain predictor variables that are manipulated (or observed) between-Ss. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.
between_covariates	Names of columns in data that contain predictor variables that are manipulated (or observed) between-Ss and are to serve as covariates in the analysis. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.
observed	Names of columns in data that are already specified in either within or between that contain predictor variables that are observed variables (i.e. not manipulated). If a single value, may be specified by name alone; if multiple values, must be specified as a .() list. The presence of observed variables affects the computation of the generalized eta-squared measure of effect size reported by ezANOVA.
diff	Names of any variables to collapse to a difference score. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list. All supplied variables must be factors, ideally with only two levels (especially if setting the reverse_diff argument to TRUE).
reverse_diff	Logical. If TRUE, triggers reversal of the difference collapse requested by diff. Take care with variables with more than 2 levels.

`type` Numeric value (either 1, 2 or 3) specifying the Sums of Squares "type" to employ when data are unbalanced (eg. when group sizes differ). `type = 2` is the default because this will yield identical ANOVA results as `type = 1` when data are balanced but `type = 2` will additionally yield various assumption tests where appropriate. When data are unbalanced, users are warned that they should give special consideration to the value of `type`. `type=3` will emulate the approach taken by popular commercial statistics packages like SAS and SPSS, but users are warned that this approach is not without criticism.

`white.adjust` Only affects behaviour if the design contains only between-Ss predictor variables. If not FALSE, the value is passed as the white.adjust argument to `Anova`, which provides heteroscedasticity correction. See `Anova` for details on possible values.

`detailed` Logical. If TRUE, returns extra information (sums of squares columns, intercept row, etc.) in the ANOVA table.

`return_aov` Logical. If TRUE, computes and returns an aov object corresponding to the requested ANOVA (useful for computing post-hoc contrasts).

Details

ANCOVA is implemented by first regressing the DV against each covariate (after collapsing the data to the means of that covariate's levels per subject) and subtracting from the raw data the fitted values from this regression (then adding back the mean to maintain scale). These regressions are computed across Ss in the case of between-Ss covariates and computed within each Ss in the case of within-Ss covariates.

Value

A list containing one or more of the following components:

`ANOVA` A data frame containing the ANOVA results.

`Mauchly's Test for Sphericity`
If any within-Ss variables with >2 levels are present, a data frame containing the results of Mauchly's test for Sphericity. Only reported for effects >2 levels because sphericity necessarily holds for effects with only 2 levels.

`Sphericity Corrections`
If any within-Ss variables are present, a data frame containing the Greenhouse-Geisser and Huynh-Feldt epsilon values, and corresponding corrected p-values.

`Levene's Test for Homogeneity`
If the design is purely between-Ss, a data frame containing the results of Levene's test for Homogeneity of variance. Note that Huynh-Feldt corrected p-values where the Huynh-Feldt epsilon >1 will use 1 as the correction epsilon.

`aov` An aov object corresponding to the requested ANOVA.

Some column names in the output data frames are abbreviated to conserve space:

DFn	Degrees of Freedom in the numerator (a.k.a. DFeffect).
DFd	Degrees of Freedom in the denominator (a.k.a. DFerror).
SSn	Sum of Squares in the numerator (a.k.a. SSeffect).

SSd	Sum of Squares in the denominator (a.k.a. SSerror).
F	F-value.
p	p-value (probability of the data given the null hypothesis).
p<.05	Highlights p-values less than the traditional alpha level of .05.
ges	Generalized Eta-Squared measure of effect size (see in references below: Bakeman, 2005).
GGe	Greenhouse-Geisser epsilon.
p[GGe]	p-value after correction using Greenhouse-Geisser epsilon.
p[GGe]<.05	Highlights p-values (after correction using Greenhouse-Geisser epsilon) less than the traditional alpha level of .
HFe	Huynh-Feldt epsilon.
p[HFe]	p-value after correction using Huynh-Feldt epsilon.
p[HFe]<.05	Highlights p-values (after correction using Huynh-Feldt epsilon) less than the traditional alpha level of .05.
W	Mauchly's W statistic

Warning

Prior to running (though after obtaining running ANCOVA regressions as described in the `details` section), `dv` is collapsed to a mean for each cell defined by the combination of `wid` and any variables supplied to `within` and/or `between` and/or `diff`. Users are warned that while convenient when used properly, this automatic collapsing can lead to inconsistencies if the pre-collapsed data are unbalanced (with respect to cells in the full design) and only the partial design is supplied to `ezANOVA`. When this is the case, use `within_full` to specify the full design to ensure proper automatic collapsing.

Author(s)

Michael A. Lawrence <mike.lwrnc@gmail.com>
Visit the ez development site at https://github.com/mike-lawrence/ez
for the bug/issue tracker and the link to the mailing list.

References

- Bakeman, R. (2005). Recommended effect size statistics for repeated measures designs. Behavior Research Methods, 37 (3), 379-384.

See Also

ezBoot, ezMixed, ezPerm, ezPlot, ezStats

Examples

```
#Read in the ANT data (see ?ANT).
data(ANT)
head(ANT)
ezPrecis(ANT)


#Run an ANOVA on the mean correct RT data.
rt_anova = ezANOVA(
    data = ANT[ANT$error==0,]
    , dv = rt
```

```
    , wid = subnum
    , within = .(cue,flank)
    , between = group
)

#Show the ANOVA and assumption tests.
print(rt_anova)

## Not run:
#Run an ANOVA on the mean correct RT data, ignoring group.
rt_anova2 = ezANOVA(
    data = ANT[ANT$error==0,]
    , dv = rt
    , wid = subnum
    , within = .(cue,flank)
)

#Show the ANOVA and assumption tests.
print(rt_anova2)

## End(Not run)

#Run a purely between-Ss ANOVA on the mean_rt data.
#NOTE use of within_full to ensure that the data are
# collapsed properly
rt_anova3 = ezANOVA(
    data = ANT[ANT$error==0,]
    , dv = rt
    , wid = subnum
    , within_full = .(cue,flank)
    , between = group
)

#Show the ANOVA and assumption tests.
print(rt_anova3)


#add a within-Ss effect to be used as a covariate
ANT$rt2 = ANT$rt + ANT$block*1000 #additive with and independent of the other predictors!

## Not run:
#Run an anova that doesn't use the covariate
rt_anova4a = ezANOVA(
    data = ANT[ANT$error==0,]
    , dv = rt2
    , wid = subnum
    , within = .(cue,flank)
    , between = group
)

#Show the ANOVA and assumption tests.
# Note loss of power to observe the within effects
print(rt_anova4a)
```

```
## End(Not run)

#Run an anova that does use the covariate
rt_anova4b = ezANOVA(
    data = ANT[ANT$error==0,]
    , dv = rt2
    , wid = subnum
    , within = .(cue,flank)
    , within_covariates = block
    , between = group
)

#Show the ANOVA and assumption tests.
# Note power to observe the within effects has returned
print(rt_anova4b)


#add a between-Ss effect to be used as a covariate
ANT$bc = as.numeric(as.character(ANT$subnum))%%10 #Note that the effect is balanced across groups
ANT$rt3 = ANT$rt + ANT$bc*1000 #additive with and independent of the other predictors!

## Not run:
#Run an anova that doesn't use the covariate
rt_anova5a = ezANOVA(
    data = ANT[ANT$error==0,]
    , dv = rt2
    , wid = subnum
    , within = .(cue,flank)
    , between = group
)

#Show the ANOVA and assumption tests.
# Note loss of power to observe the between effects
print(rt_anova5a)

## End(Not run)

#Run an anova that does use the covariate
rt_anova5b = ezANOVA(
    data = ANT[ANT$error==0,]
    , dv = rt2
    , wid = subnum
    , within = .(cue,flank)
    , between = group
    , between_covariates = bc
)

#Show the ANOVA and assumption tests.
# Note power to observe the between effects has returned
print(rt_anova5b)
```

ezBoot *Compute bootstrap resampled predictions*

Description

This function is used to compute bootstrap resampled predictions for each cell in a specified experimental design, using either cell means or mixed effects modelling to obtain predictions. The results can be visualized using ezPlot2.

Usage

```
ezBoot(
    data
    , dv
    , wid
    , within = NULL
    , between = NULL
    , resample_within = TRUE
    , iterations = 1e3
    , lmer = FALSE
    , lmer_family = gaussian
    , parallel = FALSE
    , alarm = FALSE
)
```

Arguments

data	Data frame containing the data to be analyzed.
dv	Name of the column in data that contains the dependent variable. Values in this column must be numeric.
wid	Name of the column in data that contains the variable specifying the case/Ss identifier.
within	Names of columns in data that contain predictor variables that are manipulated (or observed) within-Ss. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.
between	Names of columns in data that contain predictor variables that are manipulated (or observed) between-Ss. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.
resample_within	Logical value specifying whether to resample within each cell of the design within each wid unit. If there is only one observation per such cells, then this should be set to FALSE to avoid useless computation.
iterations	Numeric value specifying the number of bootstrap iterations to complete.
lmer	Logical. If TRUE, predictions are obtained via mixed effects modelling; if FALSE predictions are obtained via cell means.

lmer_family	When obtaining predictions via mixed effects modelling (i.e. when lmer=TRUE), you must specify the residuals family. While the bootstrap is in theory non-parametric, it may be more powerful if you specify a family that might reasonably expected to match your data. For example, if the data are binary outcomes (eg. accuracy), then use the binomial family. See lmer.
parallel	Logical. If TRUE, computation will be parallel, assuming that a parallel backend has been specified (as in library(doMC);options(cores=4);registerDoMC(). Likely only to work when running R from a unix terminal.)
alarm	Logical. If TRUE, call the alarm function when ezBoot completes.

Details

While within and between are both optional, at least one column of data must be provided to either within or between. Any numeric or character variables in data that are specified as either wid, within or between will be converted to a factor with a warning. Prior to running, dv is collapsed to a mean for each cell defined by the combination of wid, within or between.

Value

A list containing either two or three components:

fit	If predictions are obtained by mixed effects modelling, an link[lme4]{lmer} object consisting of the original mixed effects model
cells	A data frame containing predictions for each cell of the design.
boots	A data frame containing predictions for each cell of the design from each iteration of the bootstrap procedure.

Author(s)

Michael A. Lawrence <mike.lwrnc@gmail.com>
Visit the ez development site at https://github.com/mike-lawrence/ez
for the bug/issue tracker and the link to the mailing list.

See Also

link{ezANOVA}, ezMixed, ezPerm, ezPlot2, ezResample

Examples

```
#Read in the ANT data (see ?ANT).
data(ANT)
head(ANT)
ezPrecis(ANT)

#Run ezBoot on the accurate RT data
rt = ezBoot(
    data = ANT
    , dv = rt
    , wid = subnum
    , within = .(cue,flank)
```

```
    , between = group
    , iterations = 1e1 #1e3 or higher is best for publication
)

## Not run:
#plot the full design
p = ezPlot2(
    preds = rt
    , x = flank
    , split = cue
    , col = group
)
print(p)

#plot the effect of group across the flank*cue design
p = ezPlot2(
    preds = rt
    , x = flank
    , split = cue
    , diff = group
)
print(p)

#plot the flank*cue design, averaging across group
p = ezPlot2(
    preds = rt
    , x = flank
    , split = cue
)
print(p)

## End(Not run)
```

---

ezCor *Compute and plot an information-dense correlation matrix*

Description

This function provides simultaneous visualization of a correlation matrix, scatter-plot with linear fits, and univariate density plots for multiple variables.

Usage

```
ezCor(
    data
    , r_size_lims = c(10,30)
    , point_alpha = .5
    , density_height = 1
    , density_adjust = 1
    , density_colour = 'white'
```

```
    , label_size = 10
    , label_colour = 'black'
    , label_alpha = .5
    , lm_colour = 'red'
    , ci_colour = 'green'
    , ci_alpha = .5
    , test_alpha = .05
    , test_correction = 'none'
)
```

Arguments

data	Data frame containing named columns of data only.
r_size_lims	Minimum and maximum size of the text reporting the correlation coefficients. Minimum is mapped to coefficients of 0 and maximum is mapped to coefficients of 1, with the mapping proportional to r^2.
point_alpha	Transparency of the data points (1 = opaque).
density_height	Proportion of the facet height taken up by the density plots.
density_adjust	Adjusts the bandwidth of the univariate density estimator. See adjust parameter in density.
density_colour	Colour of the density plot.
label_size	Size of the variable labels on the diagonal.
label_colour	Colour of the variable labels on the diagonal.
label_alpha	Transparency of the variable labels on the diagonal (1 = opaque).
lm_colour	Colour of the fitted line.
ci_colour	Colour of the confidence interval surrounding the fitted line.
ci_alpha	Transparency of the confidence interval surrounding the fitted line (1 = opaque).
test_alpha	Type-I error rate requested for colouring of the "significant" correlation coefficients.
test_correction	Character string specifying the type of correction for multiple comparisons applied to the value specified by test_alpha. Possible values are "none", "bonferroni", and "sidak",

Value

A printable/modifiable ggplot2 object.

Author(s)

Michael A. Lawrence <mike.lwrnc@gmail.com>
Visit the ez development site at https://github.com/mike-lawrence/ez
for the bug/issue tracker and the link to the mailing list.

Examples

```
########
# Set up some fake data
########
library(MASS)
N=100

#first pair of variables
variance1=1
variance2=2
mean1=10
mean2=20
rho = .8
Sigma=matrix(
    c(
        variance1
        , sqrt(variance1*variance2)*rho
        , sqrt(variance1*variance2)*rho
        , variance2
    )
    , 2
    , 2
)
pair1=mvrnorm(N,c(mean1,mean2),Sigma,empirical=TRUE)

#second pair of variables
variance1=10
variance2=20
mean1=100
mean2=200
rho = -.4
Sigma=matrix(
    c(
        variance1
        , sqrt(variance1*variance2)*rho
        , sqrt(variance1*variance2)*rho
        , variance2
    )
    , 2
    , 2
)
pair2=mvrnorm(N,c(mean1,mean2),Sigma,empirical=TRUE)

my_data=data.frame(cbind(pair1,pair2))

########
# Now plot
########
p = ezCor(
    data = my_data
)
print(p)
```

```
#you can modify the default colours of the
##correlation coefficients as follows
library(ggplot2)
p = p + scale_colour_manual(values = c('red','blue'))
print(p)
#see the following for alternatives:
# http://had.co.nz/ggplot2/scale_manual.html
# http://had.co.nz/ggplot2/scale_hue.html
# http://had.co.nz/ggplot2/scale_brewer.html
```

`ezDesign` *Plot the balance of data in an experimental design*

Description

This function provides easy visualization of the balance of data in a data set given a specified experimental design. This function is useful for identifying missing data and other issues (see examples).

Usage

```
ezDesign(
    data
    , x
    , y
    , row = NULL
    , col = NULL
    , cell_border_size = 10
)
```

Arguments

`data`	Data frame containing the data to be visualized.
`x`	Name of the variable to plot on the x-axis.
`y`	Name of the variable to plot on the y-axis.
`row`	Name of a variable by which to split the data into facet rows.
`col`	Name of a variable by which to split the data into facet columns.
`cell_border_size`	Numeric value specifying the size of the border seperating cells (0 specifies no border)

Details

The function works by counting the number of rows in `data` in each cell of the design specified by the factorial combination of `x`, `y`, `row`, `col` variables.

Value

A printable/modifiable ggplot2 object.

Author(s)

Michael A. Lawrence <mike.lwrnc@gmail.com>
Visit the ez development site at https://github.com/mike-lawrence/ez
for the bug/issue tracker and the link to the mailing list.

See Also

ezPrecis

Examples

```
#Read in the ANT2 data (see ?ANT2).
data(ANT2)
head(ANT2)
ezPrecis(ANT2)

#toss NA trials
ANT2 = ANT2[!is.na(ANT2$rt),]

ezDesign(
    data = ANT2
    , x = trial
    , y = subnum
    , row = block
    , col = group
)
#subnum #7 is missing data from the last half of the experiment

## Not run:
ezDesign(
    data = ANT2
    , x = flank
    , y = subnum
    , row = cue
)
#again, subnum#7 has half the data as the rest

#now look at error rates, which affect the number of RTs we can use
ezDesign(
    data = ANT2[ANT2$error==0,]
    , x = flank
    , y = subnum
    , row = cue
)
#again, subnum#7 stands out because they have half the data as the rest
#also, subnum#14 has no data in any incongruent cells, suggesting that
##they made all errors in this condition
#finally, subnum#12 has virtually no data, suggesting that they mistakenly
```

```
##swapped responses

## End(Not run)
```

`ezMixed` *Compute evidence for fixed effects in an mixed effects modelling context*

Description

This function provides assessment of fixed effects and their interactions via generalized mixed effects modelling, or generalized additive mixed effects modelling for effects involving numeric predictors to account for potentially non-linear effects of such predictors. See Details section below for implementation notes.

Usage

```
ezMixed(
    data
    , dv
    , family = gaussian
    , random
    , fixed
    , covariates = NULL
    , add_q = FALSE
    , fix_gam = TRUE
    , cov_gam = TRUE
    , gam_smooth = c('s','te')
    , gam_bs = 'ts'
    , gam_k = Inf
 , use_bam = FALSE
    , alarm = FALSE
    , term_labels = NULL
    , highest = Inf
    , return_models = TRUE
    , correction = AIC
    , progress_dir = NULL
    , resume = FALSE
    , parallelism = 'none'
    , gam_args = NULL
    , mer_args = NULL
)
```

Arguments

`data`	Data frame containing the data to be analyzed.
`dv`	.() object specifying the column in `data` that contains the dependent variable. Values in this column must be numeric.

family	Family to use to represent error.
random	.() object specifying one or more columns in data that contain random effects.
fixed	.() object specifying one or more columns in data that contain fixed effects.
covariates	.() object specifying one or more columns in data that contain variables to be used as fixed effect covariates.
add_q	Logical. If TRUE, quantile values of each observation will be computed for each effect and interaction and these quantile values will be added as a fixed effect predictor. This permits investigating the effect of the fixed effect predictors specified in fixed on the shape of the distribution of residuals. Of course, this only really makes sense when there IS a distribution of residuals (i.e. not binomial data).
fix_gam	Logical. If TRUE (default), generalized additive modelling is used to evaluate the possibly-non-linear effects of numeric fixed effect predictors.
cov_gam	Logical. If TRUE (default), generalized additive modelling is used to represent the possibly-non-linear effects of numeric covariates.
gam_smooth	Vector of one or more elements that are character strings specifying the name of the smoother to use when using gam. If a list of two elements, the first element will be used when evaluating effects and interactions that include a single numeric predictor, while the second element will be used when evaluating effects and interactions that involve multiple numeric predictors.
gam_bs	Character specifying the name of the smooth term to use when using gam.
gam_k	Numeric value specifying the maximum value for k to supply to calls to gam. Higher values yield longer computation times but may better capture non-linear phenomena. If set to Inf (default), ezMixed will automatically use the maximum possible value for k given the number of unique combinations of values in the numeric predictors being evaluated. If a finite positive value is supplied, k will be set to that value or less (if the supplied k exceeds the maximum possible k for a given effect).
use_bam	Logical. If TRUE, bam is used rather than gam.
alarm	Logical. If TRUE, call the alarm function when ezMixed completes.
term_labels	Vector of one or more elements that are character strings specifying effects to explore (useful when you want only a subset of all possible effects and interactions between the predictors supplied to the fixed argument).
highest	Integer specifying the highest order interaction between the fixed effects to test. The default value, Inf, will test to the highest possible order.
return_models	Logical. If TRUE, the returned list object will also include each lmer model (can become memory intensive for complex models and/or large data sets).
correction	Name of a correction for complexity to apply (ex. AIC, BIC, etc) to each model's likelihood before computing likelihood ratios.
progress_dir	Character string specifying name of a folder to be created to store results as they are computed (to save RAM).
resume	Logical. If TRUE and a value is passed to the progress_dir argument, the progress directory will be searched for already completed effects and resume from these. Useful if a run was interrupted.

`parallelism` Character string specifying whether and how to compute models in parallel. If "none", no parallelism will be employed. If "pair", the restricted and unrestricted models for each effect will be computed in parallel (therefore using only 2 cores). If "full", then effects themselves will be computed in parallel (using all available cores). Parallelism assumes that a parallel backend has been specified (as in `library(doMC);options(cores=4);registerDoMC()`) and is likely only to work when running R from a unix terminal.

`gam_args` Single character string representing arguments to be passed to calls to `gam`.

`mer_args` Single character string representing arguments to be passed to calls to `lmer` (or `glmer` if the value to the `family` argument is not `gaussian`).

Details

Computation is achieved via `lmer`, or `gam` when the effect under evaluation includes a numeric predictor. Assessment of each effect of interest necessitates building two models: (1) a "unrestricted" model that contains the effect of interest plus any lower order effects and (2) a "restricted" model that contains only the lower order effects (thus "restricting" the effect of interest to zero). These are then compared by means of a likelihood ratio, which needs to be corrected to account for the additional complexity of the unrestricted model relative to the restricted model. The default applied correction is Akaike's Information Criterion (AIC), which in the context of mixed effects models has been demonstrated to be asymptotically equivalent to cross-validation, a gold-standard technique for ensuring that model comparisons optimize prediction of new data.

The complexity-corrected likelihood ratio returned by `ezMixed` is represented on the log-base-2 scale, which has the following convenient properties:

- (1) Resulting values can be discussed as representing "bits of evidence" for or against the evaluated effect.
- (2) The bits scale permits easy representation of both very large and very small likelihood ratios.
- (3) The bits scale represents equivalent evidence between the restricted and unrestricted models by a value of 0.
- (4) The bits scale represents ratios favoring the restricted model symmetrically to those favoring the unrestricted model. That is, say one effect obtains a likelihood ratio of 8, and another effect obtains a likelihood ratio of 0.125; both ratios indicate the same degree of imbalance of evidence (8:1 and 1:8) and on the bits scale they faithfully represent this symmetry as values 3 and -3, respectively.

Value

A list with the following elements:

`summary` A data frame summarizing the results, including whether warnings or errors occurred during the assessment of each effect and the bits of evidence associated with each.

`formulae` A list of lists, each named for an effect and containing two elements named "unrestricted" and "restricted", which in turn contain the right-hand-side formulae used to fit the unrestricted and restricted models, respectively.

errors	A list similar to formulae, but instead storing errors encountered in fitting each model.
warnings	A list similar to formulae, but instead storing warnings encountered in fitting each model.
models	(If requested by setting return_models=TRUE) A list similar to formulae but instead storing each fitted model.

Author(s)

Michael A. Lawrence <mike.lwrnc@gmail.com>
Visit the ez development site at https://github.com/mike-lawrence/ez
for the bug/issue tracker and the link to the mailing list.

References

- Glover S, Dixon P. (2004) Likelihood ratios: a simple and flexible statistic for empirical psychologists. Psychonomic Bulletin and Review, 11 (5), 791-806.
- Fang, Y. (2011). Asymptotic equivalence between cross-validations and Akaike information criteria in mixed-effects models. Journal of the American Statistical Association, 9, 15-21.

See Also

lmer, glmer, gam, ezMixedProgress, ezPredict, ezPlot2

Examples

```
#Read in the ANT data (see ?ANT).
data(ANT)
head(ANT)
ezPrecis(ANT)

#Run ezMixed on the accurate RT data
rt = ezMixed(
    data = ANT[ANT$error==0,]
    , dv = .(rt)
    , random = .(subnum)
    , fixed = .(cue,flank,group)
)
print(rt$summary)

## Not run:
#Run ezMixed on the error rate data
er = ezMixed(
    data = ANT
    , dv = .(error)
    , random = .(subnum)
    , fixed = .(cue,flank,group)
    , family = 'binomial'
)
print(er$summary)
```

```
## End(Not run)
```

`ezMixedProgress` *Retrieve information saved to file by a call to ezMixed*

Description

When running `ezMixed` with a value supplied to the `progress_dir` argument, summary results are saved to file. `ezMixedProgress` retrieves those results, even from partial or discontinued runs.

Usage

```
ezMixedProgress(
    progress_dir
    , return_models = TRUE
)
```

Arguments

`progress_dir`	Character string specifying the name of the progress directory. (Should match the string supplied as the value to the `progress_dir` argument in the original call to `ezMixed`)
`return_models`	Logical. If TRUE, the returned list object will also include each lmer model (can become memory intensive for complex models and/or large data sets).

Value

A list with the following elements:

`summary`	A data frame summarizing the results, including whether warnings or errors occurred during the assessment of each effect, raw natural-log likelihood of the unrestricted and restricted models (RLnLu and RLnLr, respectively), degrees of freedom of the unrestricted and restricted models (DFu and DFr, respectively), and log-base-10 likelihood ratios corrected via AIC and BIC (L10LRa and L10LRb, respectively)
`formulae`	A list of lists, each named for an effect and containing two elements named "unrestricted" and "restricted", which in turn contain the right-hand-side formulae used to fit the unrestricted and restricted models, respectively.
`errors`	A list similar to `formulae`, but instead storing errors encountered in fitting each model.
`warnings`	A list similar to `formulae`, but instead storing warnings encountered in fitting each model.
`models`	(If requested by setting `return_models=TRUE`) A list similar to `formulae` but instead storing each fitted model.

Author(s)

Michael A. Lawrence <mike.lwrnc@gmail.com>
Visit the ez development site at https://github.com/mike-lawrence/ez
for the bug/issue tracker and the link to the mailing list.

See Also

ezMixed

Examples

```
## Not run:
#Read in the ANT data (see ?ANT).
data(ANT)
head(ANT)
ezPrecis(ANT)

#Run ezMixed on the accurate RT data
rt_mix = ezMixed(
    data = ANT[ANT$error==0,]
    , dv = .(rt)
    , random = .(subnum)
    , fixed = .(cue,flank,group)
    , progress_dir = 'rt_mix'
)

rt_mix = ezMixedProgress('rt_mix')
print(rt_mix$summary)

## End(Not run)
```

---

ezPerm *Perform a factorial permutation test*

Description

This function provides easy non-parametric permutation test analysis of data from factorial experiments, including purely within-Ss designs (a.k.a. "repeated measures"), purely between-Ss designs, and mixed within-and-between-Ss designs.

Usage

```
ezPerm(
    data
    , dv
    , wid
    , within = NULL
```

```
    , between = NULL
    , perms = 1e3
    , parallel = FALSE
    , alarm = FALSE
)
```

Arguments

data	Data frame containing the data to be analyzed.
dv	Name of the column in data that contains the dependent variable. Values in this column must be numeric.
wid	Name of the column in data that contains the variable specifying the case/Ss identifier.
within	Names of columns in data that contain predictor variables that are manipulated (or observed) within-Ss. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.
between	Names of columns in data that contain predictor variables that are manipulated (or observed) between-Ss. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.
perms	An integer > 0 specifying the number of permutations to compute.
parallel	Logical. If TRUE, computation will be parallel, assuming that a parallel backend has been specified (as in `library(doMC);options(cores=4);registerDoMC()`. Likely only to work when running R from a unix terminal.)
alarm	Logical. If TRUE, call the alarm function when ezPerm completes.

Value

A data frame containing the permutation test results.

Warning

ezPerm() is a work in progress. Under the current implementation, only main effects may be trusted.

Author(s)

Michael A. Lawrence <mike.lwrnc@gmail.com>
Visit the ez development site at https://github.com/mike-lawrence/ez
for the bug/issue tracker and the link to the mailing list.

See Also

link{ezANOVA}, ezBoot, ezMixed

Examples

```
library(plyr)
#Read in the ANT data (see ?ANT).
data(ANT)
head(ANT)
ezPrecis(ANT)

#Compute some useful statistics per cell.
cell_stats = ddply(
    .data = ANT
    , .variables = .( subnum , group , cue , flank )
    , .fun = function(x){
        #Compute error rate as percent.
        error_rate = mean(x$error)*100
        #Compute mean RT (only accurate trials).
        mean_rt = mean(x$rt[x$error==0])
        #Compute SD RT (only accurate trials).
        sd_rt = sd(x$rt[x$error==0])
        to_return = data.frame(
            error_rate = error_rate
            , mean_rt = mean_rt
            , sd_rt = sd_rt
        )
        return(to_return)
    }
)

#Compute the grand mean RT per Ss.
gmrt = ddply(
    .data = cell_stats
    , .variables = .( subnum , group )
    , .fun = function(x){
        to_return = data.frame(
            mrt = mean(x$mean_rt)
        )
        return(to_return)
    }
)

#Run a purely between-Ss permutation test on the mean_rt data.
mean_rt_perm = ezPerm(
    data = gmrt
    , dv = mrt
    , wid = subnum
    , between = group
    , perms = 1e1 #1e3 or higher is best for publication
)

#Show the Permutation test.
print(mean_rt_perm)
```

ezPlot *Plot data from a factorial experiment*

Description

This function provides easy visualization of any given user-requested effect from factorial experiments, including purely within-Ss designs (a.k.a. "repeated measures"), purely between-Ss designs, and mixed within-and-between-Ss designs. By default, Fisher's Least Significant Difference is computed to provide error bars that facilitate visual post-hoc multiple comparisons (see Warning section below).

Usage

```
ezPlot(
    data
    , dv
    , wid
    , within = NULL
    , within_full = NULL
    , within_covariates = NULL
    , between = NULL
    , between_full = NULL
    , between_covariates = NULL
    , x
    , do_lines = TRUE
    , do_bars = TRUE
    , bar_width = NULL
    , bar_size = NULL
    , split = NULL
    , row = NULL
    , col = NULL
    , to_numeric = NULL
    , x_lab = NULL
    , y_lab = NULL
    , split_lab = NULL
    , levels = NULL
    , diff = NULL
    , reverse_diff = FALSE
    , type = 2
    , dv_levs = NULL
    , dv_labs = NULL
    , y_free = FALSE
    , print_code = FALSE
)
```

Arguments

data	Data frame containing the data to be analyzed. OR, if multiple values are specified in dv, a list with as many element as values specified in dv, each element specifying a data frame for each dv in sequence.
dv	.() object specifying the column in data that contains the dependent variable. Values in this column should be of the numeric class. Multiple values will yield a plot with dv mapped to row.
wid	.() object specifying the column in data that contains the variable specifying the case/Ss identifier. Values in this column will be converted to factor class if necessary.
within	Names of columns in data that contain predictor variables that are manipulated (or observed) within-Ss. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.
within_full	Same as within, but intended to specify the full within-Ss design in cases where the data have not already been collapsed to means per condition specified by within and when within only specifies a subset of the full design.
within_covariates	Names of columns in data that contain predictor variables that are manipulated (or observed) within-Ss and are to serve as covariates in the analysis. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.
between	Names of columns in data that contain predictor variables that are manipulated (or observed) between-Ss. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.
between_full	Same as between, but must specify the full set of between-Ss variables if between specifies only a subset of the design.
between_covariates	Names of columns in data that contain predictor variables that are manipulated (or observed) between-Ss and are to serve as covariates in the analysis. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.
x	.() object specifying the variable to plot on the x-axis.
do_lines	Logical. If TRUE, lines will be plotted connecting groups of points.
do_bars	Logical. If TRUE, error bars will be plotted.
bar_width	Optional numeric value specifying custom widths for the error bar hat.
bar_size	Optional numeric value or vector specifying custom size of the error bars.
split	Optional .() object specifying a variable by which to split the data into different shapes/colors (and line types, if do_lines==TRUE).
row	Optional .() object specifying a variable by which to split the data into rows.
col	Optional .() object specifying a variable by which to split the data into columns.
to_numeric	Optional .() object specifying any variables that need to be converted to the numeric class before plotting.
x_lab	Optional character string specifying the x-axis label.

y_lab	Optional character string specifying the y-axis label.
split_lab	Optional character string specifying the key label.
levels	Optional named list where each item name matches a factored column in data that needs either reordering of levels, renaming of levels, or both. Each item should be a list containing named elements new_order or new_names or both.
diff	Optional .() object specifying a 2-level within-Ss varbiable to collapse to a difference score.
reverse_diff	Logical. If TRUE, triggers reversal of the difference collapse requested by diff.
type	Numeric value (either 1, 2 or 3) specifying the Sums of Squares "type" to employ when data are unbalanced (eg. when group sizes differ). See ezANOVA for details.
dv_levs	Optional character vector specifying the factor ordering of multiple values specified in dv.
dv_labs	Optional character vector specifying new factor labels for each of the multiple values specified in dv.
y_free	Logical. If TRUE, then rows will permit different y-axis scales.
print_code	Logical. If TRUE, the code for creating the ggplot2 plot object is printed and the data to be plotted is returned instead of the plot itself.

Details

ANCOVA is implemented by first regressing the DV against each covariate (after collapsing the data to the means of that covariate's levels per subject) and subtracting from the raw data the fitted values from this regression (then adding back the mean to maintain scale). These regressions are computed across Ss in the case of between-Ss covariates and computed within each Ss in the case of within-Ss covariates.

Fisher's Least Significant Difference is computed as sqrt(2)*qt(.975,DFd)*sqrt(MSd/N), where N is taken as the mean N per group in cases of unbalanced designs.

Value

If print_code is FALSE, printable/modifiable ggplot2 object is returned. If print_code is TRUE, the code for creating the ggplot2 plot object is printed and the data to be plotted is returned instead of the plot itself.

Warnings

Prior to running (though after obtaining running ANCOVA regressions as described in the details section), dv is collapsed to a mean for each cell defined by the combination of wid and any variables supplied to within and/or between and/or diff. Users are warned that while convenient when used properly, this automatic collapsing can lead to inconsistencies if the pre-collapsed data are unbalanced (with respect to cells in the full design) and only the partial design is supplied to ezANOVA. When this is the case, use within_full to specify the full design to ensure proper automatic collapsing.

The default error bars are Fisher's Least Significant Difference for the plotted effect, facilitating visual post-hoc multiple comparisons. To obtain accurate FLSDs when only a subset of the full

between-Ss design is supplied to `between`, the full design must be supplied to `between_full`. Also note that in the context of mixed within-and-between-Ss designs, the computed FLSD bars can only be used for within-Ss comparisons.

Author(s)

Michael A. Lawrence <mike.lwrnc@gmail.com>
Visit the ez development site at https://github.com/mike-lawrence/ez
for the bug/issue tracker and the link to the mailing list.

See Also

ezANOVA, ezStats

Examples

```
#Read in the ANT data (see ?ANT).
data(ANT)
head(ANT)
ezPrecis(ANT)


## Not run:
#Run an ANOVA on the mean correct RT data.
mean_rt_anova = ezANOVA(
    data = ANT[ANT$error==0,]
    , dv = .(rt)
    , wid = .(subnum)
    , within = .(cue,flank)
    , between = .(group)
)

#Show the ANOVA and assumption tests.
print(mean_rt_anova)

## End(Not run)

#Plot the main effect of group.
group_plot = ezPlot(
    data = ANT[ANT$error==0,]
    , dv = .(rt)
    , wid = .(subnum)
    , between = .(group)
    , x = .(group)
    , do_lines = FALSE
    , x_lab = 'Group'
    , y_lab = 'RT (ms)'
)

#Show the plot.
print(group_plot)
```

```
#tweak the plot
# group_plot = group_plot +
# theme(
#     panel.grid.major = element_blank()
#     , panel.grid.minor = element_blank()
# )
# print(group_plot)



#use the "print_code" argument to print the
# code for creating the plot and return the
# data to plot. This is useful when you want
# to learn how to create plots from scratch
# (which can in turn be useful when you can't
# get a combination of ezPlot and tweaking to
# achieve what you want)
group_plot_data = ezPlot(
    data = ANT[ANT$error==0,]
    , dv = .(rt)
    , wid = .(subnum)
    , between = .(group)
    , x = .(group)
    , do_lines = FALSE
    , x_lab = 'Group'
    , y_lab = 'RT (ms)'
    , print_code = TRUE
)


#Re-plot the main effect of group, using the levels
##argument to re-arrange/rename levels of group
group_plot = ezPlot(
    data = ANT[ANT$error==0,]
    , dv = .(rt)
    , wid = .(subnum)
    , between = .(group)
    , x = .(group)
    , do_lines = FALSE
    , x_lab = 'Group'
    , y_lab = 'RT (ms)'
    , levels = list(
        group = list(
            new_order = c('Treatment','Control')
            , new_names = c('Treatment\nGroup','Control\nGroup')
        )
    )
)

#Show the plot.
print(group_plot)
```

```
#Plot the cue*flank interaction.
cue_by_flank_plot = ezPlot(
    data = ANT[ANT$error==0,]
    , dv = .(rt)
    , wid = .(subnum)
    , within = .(cue,flank)
    , x = .(flank)
    , split = .(cue)
    , x_lab = 'Flanker'
    , y_lab = 'RT (ms)'
    , split_lab = 'Cue'
)

#Show the plot.
print(cue_by_flank_plot)


#Plot the cue*flank interaction by collapsing the cue effect to
##the difference between None and Double
cue_by_flank_plot2 = ezPlot(
    data = ANT[ ANT$error==0 & (ANT$cue %in% c('None','Double')) ,]
    , dv = .(rt)
    , wid = .(subnum)
    , within = .(flank)
    , diff = .(cue)
    , reverse_diff = TRUE
    , x = .(flank)
    , x_lab = 'Flanker'
    , y_lab = 'RT Effect (None - Double, ms)'
)

#Show the plot.
print(cue_by_flank_plot2)



#Plot the group*cue*flank interaction.
group_by_cue_by_flank_plot = ezPlot(
    data = ANT[ANT$error==0,]
    , dv = .(rt)
    , wid = .(subnum)
    , within = .(cue,flank)
    , between = .(group)
    , x = .(flank)
    , split = .(cue)
    , col = .(group)
    , x_lab = 'Flanker'
    , y_lab = 'RT (ms)'
    , split_lab = 'Cue'
)

#Show the plot.
print(group_by_cue_by_flank_plot)
```

```
#Plot the group*cue*flank interaction in both error rate and mean RT.
group_by_cue_by_flank_plot_both = ezPlot(
    data = list(
        ANT
        , ANT[ANT$error==0,]
    )
    , dv = .(error,rt)
    , wid = .(subnum)
    , within = .(cue,flank)
    , between = .(group)
    , x = .(flank)
    , split = .(cue)
    , col = .(group)
    , x_lab = 'Flanker'
    , split_lab = 'Cue'
    , dv_labs = c('ER (%)', 'RT (ms)')
    , y_free = TRUE
)

#Show the plot.
print(group_by_cue_by_flank_plot_both)
```

---

ezPlot2 *Plot bootstrap predictions and confidence intervals*

Description

This function provides easy visualization of any given user-requested effect from the bootstrap predictions computed by ezPredict or ezBoot.

Usage

```
ezPlot2(
    preds
    , CI = .95
    , x = NULL
    , split = NULL
    , row = NULL
    , col = NULL
    , do_lines = TRUE
    , ribbon = FALSE
    , CI_alpha = .5
    , point_alpha = .8
    , line_alpha = .8
    , bar_width = NULL
```

```
    , to_numeric = NULL
    , x_lab = NULL
    , y_lab = NULL
    , split_lab = NULL
    , levels = NULL
    , diff = NULL
    , reverse_diff = NULL
    , y_free = FALSE
    , alarm = FALSE
    , do_plot = TRUE
    , print_code = FALSE
    , parallel = FALSE
)
```

Arguments

preds	An list object resulting from a call to ezPredict or ezBoot.
CI	Numeric vector of one or more confidence levels to use for plotting error bars. If plotting multiple confidence regions, it is suggested that an equal number of different values are supplied to the bar_width argument for differentiation.
x	Name of the variable to plot on the x-axis.
split	Name of a variable by which to split the data into different shapes/colors (and line types, if do_lines==TRUE).
row	Name of a variable by which to split the data into facet rows.
col	Name of a variable by which to split the data into facet columns.
do_lines	Logical. If TRUE, lines will be plotted connecting groups of points.
ribbon	Logical. If TRUE, a ribbon will be plotted instead of error bars (and no points will actually be plotted, just lines).
CI_alpha	Numeric value between 0 and 1 specifying the opacity of the CI bars/ribbon.
point_alpha	Numeric value between 0 and 1 specifying the opacity of the plotted points.
line_alpha	Numeric value between 0 and 1 specifying the opacity of the plotted lines.
bar_width	Numeric value or vector specifying custom widths for the error bar hat. Must either have a length of 1, or the same length as CI.
to_numeric	Names of any variables that need to be converted to the numeric class before plotting. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.
x_lab	Character string specifying the x-axis label.
y_lab	Character string specifying the y-axis label.
split_lab	Character string specifying the key label.
levels	Named list where each item name matches a factored column in data that needs either reordering of levels, renaming of levels, or both. Each item should be a list containing named elements new_order or new_names or both.

diff	Names of any variables to collapse to a difference score. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list. All supplied variables must be factors, ideally with only two levels (especially if setting the reverse_diff argument to TRUE).
reverse_diff	Logical. If TRUE, triggers reversal of the difference collapse requested by diff. Take care with variables with more than 2 levels.
y_free	Logical. If TRUE, then rows will permit different y-axis scales.
alarm	Logical. If TRUE, call the alarm function when ezPlot2 completes (useful for plots that take a long time).
do_plot	Logical. If TRUE, no plot will be produced but instead a data frame containing point predictions and confidence limits will be returned.
print_code	Logical. If TRUE, the code for creating the ggplot2 plot object is printed and the data to be plotted is returned instead of the plot itself.
parallel	Logical. If TRUE, computation will be parallel, assuming that a parallel backend has been specified (as in library(doMC);options(cores=4);registerDoMC(). Likely only to work when running R from a unix terminal.)

Value

If do_plot is TRUE (default) and print_code if FALSE (default), a printable/modifiable ggplot2 object representing the predictions and confidence intervals. If do_plot is FALSE or print_code is TRUE, a list containing the cell predictions and bootstrapped CIs is returned.

Author(s)

Michael A. Lawrence <mike.lwrnc@gmail.com>
Visit the ez development site at https://github.com/mike-lawrence/ez
for the bug/issue tracker and the link to the mailing list.

See Also

ezBoot, ezMixed, ezPredict

Examples

```
#see example in ezPredict documentation
```

---

ezPrecis *Obtain a structure summary of a given data frame*

Description

This function provides a structure summary of a given data frame.

Usage

```
ezPrecis(
    data
    , transpose = TRUE
)
```

Arguments

data	Data frame containing the data to be analyzed.
transpose	Logical. If TRUE (default), triggers tranposition of the resulting summary data frame (useful when there are many columns in the original data frame, leading the untransposed summary data frame to wrap).

Details

This function was inspired by the whatis() function from the YaleToolkit package.

Value

A data frame containing the descriptive information about each column in the specified data frame:

type	This row indicates the type of data R thinks is in each column. Recall that when R imports data to a data frame, each column is given a label that indicates what type of information is in that column (character, numeric, or a factor data).
missing	This row reports a count of the number of missing values in each column.
unique	This row reports a count of the number of unique values in each column.
min	This row reports the minimum value found in each column. If the column data is numeric this is straightforward. If the column data is factored, the first level is reported. If the column data is character, the alphabetically first string is reported.
max	This row reports the maximum value found in each column. If the column data is numeric this is straightforward. If the column data is factored, the last level is reported. If the column data is character, the alphabetically last string is reported.

Author(s)

Michael A. Lawrence <mike.lwrnc@gmail.com>
Visit the ez development site at https://github.com/mike-lawrence/ez
for the bug/issue tracker and the link to the mailing list.

See Also

ezDesign

Examples

```
#Read in the ANT2 data (see ?ANT2).
data(ANT2)
head(ANT2)

#Show a summary of the ANT2 data.
ezPrecis(ANT2)
```

`ezPredict` *Compute predicted values from the fixed effects of a mixed effects model*

Description

This function computes the predicted values from the fixed effects of a mixed effects model.

Usage

```
ezPredict(
    fit
    , to_predict = NULL
    , numeric_res = 0
    , boot = TRUE
    , iterations = 1e3
    , zero_intercept_variance = FALSE
)
```

Arguments

`fit`	Fitted `lmer` object.
`to_predict`	Optional data frame containing the fixed effects design to predict. If absent, the function will assume that the full design from the provided fitted model is requested.
`numeric_res`	Integer value specifying the sampling resolution of any numeric fixed effect. Has no effect if non-NULL value supplied to `to_predict`. If `to_predict` is null and a numeric fixed effect is encountered in the fitted model, then predictions will be obtained at this many evenly spaced intervals between the minimum and maximum values in the original fitted data. The default value, 0, obtains predictions for each unique value found in the original data frame.
`boot`	Logical. If TRUE (default), bootstrapping will be used to generate sample predictions.
`iterations`	Integer value specifying the number of bootstrap iterations to employ if `boot==TRUE`.
`zero_intercept_variance`	Logical. If TRUE (default), bootstrap samples will be obtained after setting the intercept variance and covariances to zero. This makes sense only when, prior to fitting the model, the predictor variables were set up with contrasts that make the

intercept orthogonal to effects of interest (e.g. `contr.sum` or `contr.helmert`). This is useful to visualize cell means with confidence intervals that (roughly) can speak to differences between cells. However, it can be the case that even after zeroing the intercept variance, the resultant CIs on the raw cell values are deceptively large for comparing cells. So, if there appears to be no difference between cells, it is best to nonetheless re-visualize after collapsing the pair of cells of interest to a difference score (using the `diff` argument to `ezPlot2`).

Value

A data frame containing the prediction value (and estimated variance of this value) for each cell in the fixed effects design.

Author(s)

Michael A. Lawrence <mike.lwrnc@gmail.com>
Visit the ez development site at https://github.com/mike-lawrence/ez
for the bug/issue tracker and the link to the mailing list.

See Also

`ezMixed`, `ezPlot2`

Examples

```
library(lme4)

#Read in the ANT data (see ?ANT).
data(ANT)
head(ANT)

#fit a mixed effects model to the rt data
rt_fit = lmer(
    formula = rt ~ cue*flank*group + (1|subnum)
    , data = ANT[ANT$error==0,]
)

#obtain the predictions from the model
rt_preds = ezPredict(
    fit = rt_fit
)


#visualize the predictions
ezPlot2(
    preds = rt_preds
    , x = flank
    , row = cue
    , col = group
    , y_lab = 'RT (ms)'
)
```

`ezResample` *Resample data from a factorial experiment*

Description

This function resamples data (useful when bootstrapping and used by `ezBoot`).

Usage

```
ezResample(
    data
    , wid
    , within = NULL
    , between = NULL
    , resample_within = FALSE
    , resample_between = TRUE
    , check_args = TRUE
)
```

Arguments

`data`	Data frame containing the data to be analyzed.
`wid`	.() object specifying the column in `data` that contains the variable specifying the case/Ss identifier.
`within`	Optional .() object specifying one or more columns in `data` that contain predictor variables that are manipulated (or observed) within-Ss.
`between`	Optional .() object specifying one or more columns in `data` that contain predictor variables that are manipulated (or observed) between-Ss.
`resample_within`	Logical. If TRUE, and if there are multiple observations per subject within each cell of the design specified by the factorial combination of variables supplied to `within` and `between`, then these observations-within-cells are resampled with replacement.
`resample_between`	Logical. If TRUE (default), levels of `wid` are resampled.
`check_args`	Users should leave this as its default (TRUE) value. This argument is intended for internal use only.

Value

A data frame consisting of the resampled data

Author(s)

Michael A. Lawrence <mike.lwrnc@gmail.com>
Visit the ez development site at https://github.com/mike-lawrence/ez
for the bug/issue tracker and the link to the mailing list.

See Also

ezBoot

Examples

```
library(plyr)
#Read in the ANT data (see ?ANT).
data(ANT)
head(ANT)
ezPrecis(ANT)


#Bootstrap the within-cell variances
var_boots = ldply(
    .data = 1:1e1 #1e3 or higher should be used for publication
    , .fun = function(x){
        this_resample = ezResample(
            data = ANT[ANT$error==0,]
            , wid = .(subnum)
            , within = .(cue,flank)
            , between = .(group)
        )
        cell_vars = ddply(
            .data = idata.frame(this_resample)
            , .variables = .(subnum,cue,flank,group)
            , .fun = function(x){
                to_return = data.frame(
                    value = var(x$rt)
                )
                return(to_return)
            }
        )
        mean_cell_vars = ddply(
            .data = idata.frame(cell_vars)
            , .variables = .(cue,flank,group)
            , .fun = function(x){
                to_return = data.frame(
                    value = mean(x$value)
                )
                return(to_return)
            }
        )
        mean_cell_vars$iteration = x
        return(mean_cell_vars)
    }
    , .progress = 'time'
)
```

ezStats *Compute descriptive statistics from a factorial experiment*

Description

This function provides easy computation of descriptive statistics (between-Ss means, between-Ss SD, Fisher's Least Significant Difference) for data from factorial experiments, including purely within-Ss designs (a.k.a. "repeated measures"), purely between-Ss designs, and mixed within-and-between-Ss designs.

Usage

```
ezStats(
    data
    , dv
    , wid
    , within = NULL
    , within_full = NULL
    , within_covariates = NULL
    , between = NULL
    , between_full = NULL
    , between_covariates = NULL
    , diff = NULL
    , reverse_diff = FALSE
    , type = 2
    , check_args = TRUE
)
```

Arguments

data	Data frame containing the data to be analyzed.
dv	Name of the column in data that contains the dependent variable. Values in this column must be numeric.
wid	Name of the column in data that contains the variable specifying the case/Ss identifier.
within	Names of columns in data that contain predictor variables that are manipulated (or observed) within-Ss. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.
within_full	Same as within, but intended to specify the full within-Ss design in cases where the data have not already been collapsed to means per condition specified by within and when within only specifies a subset of the full design.
within_covariates	Names of columns in data that contain predictor variables that are manipulated (or observed) within-Ss and are to serve as covariates in the analysis. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.

`between` Names of columns in `data` that contain predictor variables that are manipulated (or observed) between-Ss. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.

`between_full` Same as `between`, but must specify the full set of between-Ss variables if `between` specifies only a subset of the design.

`between_covariates` Names of columns in `data` that contain predictor variables that are manipulated (or observed) between-Ss and are to serve as covariates in the analysis. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list.

`diff` Names of any variables to collapse to a difference score. If a single value, may be specified by name alone; if multiple values, must be specified as a .() list. All supplied variables must be factors, ideally with only two levels (especially if setting the `reverse_diff` argument to TRUE).

`reverse_diff` Logical. If TRUE, triggers reversal of the difference collapse requested by `diff`. Take care with variables with more than 2 levels.

`type` Numeric value (either `1`, `2` or `3`) specifying the Sums of Squares "type" to employ when data are unbalanced (eg. when group sizes differ). `type = 2` is the default because this will yield identical ANOVA results as `type = 1` when data are balanced but `type = 2` will additionally yield various assumption tests where appropriate. When data are unbalanced, users are warned that they should give special consideration to the value of `type`. `type=3` will emulate the approach taken by popular commercial statistics packages like SAS and SPSS, but users are warned that this approach is not without criticism.

`check_args` Users should leave this as its default (TRUE) value. This argument is intended for internal use only.

Details

ANCOVA is implemented by first regressing the DV against each covariate (after collapsing the data to the means of that covariate's levels per subject) and subtracting from the raw data the fitted values from this regression (then adding back the mean to maintain scale). These regressions are computed across Ss in the case of between-Ss covariates and computed within each Ss in the case of within-Ss covariates.

Fisher's Least Significant Difference is computed as sqrt(2)*qt(.975,DFd)*sqrt(MSd/N), where N is taken as the mean N per group in cases of unbalanced designs.

Value

A data frame containing the descriptive statistics for the requested effect. N = number of Ss per cell. Mean = between-Ss mean. SD = between-Ss SD. FLSD = Fisher's Least Significant Difference.

Warnings

Prior to running (though after obtaining running ANCOVA regressions as described in the `details` section), `dv` is collapsed to a mean for each cell defined by the combination of `wid` and any variables supplied to `within` and/or `between` and/or `diff`. Users are warned that while convenient when used

properly, this automatic collapsing can lead to inconsistencies if the pre-collapsed data are unbalanced (with respect to cells in the full design) and only the partial design is supplied to ezANOVA. When this is the case, use within_full to specify the full design to ensure proper automatic collapsing.

The descriptives include Fisher's Least Significant Difference for the plotted effect, facilitating visual post-hoc multiple comparisons. To obtain accurate FLSDs when only a subset of the full between-Ss design is supplied to between, the full design must be supplied to between_full. Also note that in the context of mixed within-and-between-Ss designs, the computed FLSD values can only be used for within-Ss comparisons.

Author(s)

Michael A. Lawrence <mike.lwrnc@gmail.com>
Visit the ez development site at https://github.com/mike-lawrence/ez
for the bug/issue tracker and the link to the mailing list.

See Also

ezANOVA, ezPlot

Examples

```
#Read in the ANT data (see ?ANT).
data(ANT)
head(ANT)
ezPrecis(ANT)


#Run an ANOVA on the mean correct RT data.
mean_rt_anova = ezANOVA(
    data = ANT[ANT$error==0,]
    , dv = rt
    , wid = subnum
    , within = .(cue,flank)
    , between = group
)

#Show the ANOVA and assumption tests.
print(mean_rt_anova)

#Compute descriptives for the main effect of group.
group_descriptives = ezStats(
    data = ANT[ANT$error==0,]
    , dv = rt
    , wid = subnum
    , between = .(group)
)

#Show the descriptives.
print(group_descriptives)
```

Index