Package: LMERConvenienceFunctions (via r-universe)

August 25, 2024

Type Package

Title Model Selection and Post-Hoc Analysis for (G)LMER Models

Version 3.0

Date 2020-09-30

Author Antoine Tremblay, Statistics Canada, and Johannes Ransijn, University of Copenhagen

Maintainer ``Antoine Tremblay, Statistics Canada'' `<trea26@gmail.com>`

Description The main function of the package is to perform backward selection of fixed effects, forward fitting of the random effects, and post-hoc analysis using parallel capabilities. Other functionality includes the computation of ANOVAs with upper- or lower-bound p-values and R-squared values for each model term, model criticism plots, data trimming on model residuals, and data visualization. The data to run examples is contained in package LCF_data.

Depends lme4

Imports Matrix, LCFdata, fields, mgcv, parallel, methods, graphics

License GPL-2

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2020-10-06 10:10:06 UTC

Contents

`LMERConvenienceFunctions-package`

Model Selection and Post-Hoc Analysis for (G)LMER Models

Description

The main function of the package is to perform backward selection of fixed effects, forward fitting of the random effects, and post-hoc analyses using parallel capabilities. Other functionality includes the computation of ANOVAs with upper- or lower-bound p-values and R-squared values for each model term, model criticism plots, data trimming on model residuals, and data visualization. The data to run examples is contained in package `LCF_data`.

Details

Package:	LMERConvenienceFunctions
Type:	Package
Version:	3.0
Date:	2020-09-27
License:	GPL-2
LazyLoad:	yes

Author(s)

Antoine Tremblay, Statistics Canada, and Johannes Ransijn, University of Copenhagen

Maintainer: "Antoine Tremblay, Statistics Canada" <trea26@gmail.com>

References

Baayen, R.H. (2008). *Analyzing Linguistic Data. A Practical Introduction to Statistics Using R*. Cambridge, UK: Cambridge University Press.

Baayen, R.H., Davidson, D.J. and Bates, D.M. (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, *59*, 390–412.

Newman, A.J., Tremblay, A., Nichols, E.S., Neville, H.J., and Ullman, M.T. (2012). The Influence of Language Proficiency on Lexical Semantic Processing in Native and Late Learners of English. *Journal of Cognitive Neuroscience*, *25*, 1205–1223.

Newman, A.J., Tremblay, A., Neville, H.J., and Ullman, M.T. (In preparation). The relationship between proficiency and ERP components evoked by grammatical violations in native and late learners of English.

Pinheiro, J.C. and Bates, D.M. (2000). *Mixed Effects Models in S and S-Plus*. New York: Springer.

Quene, H., & van den Bergh, H. (2008). Examples of mixed-effects modeling with crossed random effects and with binomial data. *Journal of Memory and Language*, 59, 413–425. doi: 10.1016/j.jml.2008.02.002.

Symonds, M.R.E and Moussalli, A. (2011). *A brief guide to model selection, multimodel inference and model averaging in behavioural ecology using Akaike's information criterion. Behavioral Ecology and Sociobiology*, 65, 13–21. doi: 10.1007/s00265-010-1037-6

Tremblay, Antoine. (2009). *Processing Advantages of Lexical Bundles: Evidence from Self-paced Reading, Word and Sentence Recall, and Free Recall with Event-related Brain Potential Recordings*. Ph.D. Dissertation. University of Alberta, Edmonton, Canada.

Tremblay, A. and Tucker B. V. (2011). The Effects of N-gram Probabilistic Measures on the Processing and Production of Four-word Sequences. *The Mental Lexicon*, *6(2)*, 302–324.

See Also

`bfFixefLMER_F.fnc`; `bfFixefLMER_t.fnc`; `ffRanefLMER.fnc`; `fitLMER.fnc`; `mcposthoc.fnc`; `summary.mcposthoc`; `pamer.fnc`; `mcp.fnc`; `relLik`; `romr.fnc`; `plotLMER.fnc`; `plotLMER3d.fnc`; `plotDensity3d.fnc`; `plotRaw3d.fnc`; `perSubjectTrim.fnc`; `cn`; `f`; `cd`; `cdf`; `cdup`.

Examples

```
## Not run:
  #############################################
  #            Load and format data.          #
  #############################################
  library(LCFdata)
  data(eeg)

  # restrict to electrode Fz and 80--180 ms window
  eeg <- eeg[eeg$Time >= 80 & eeg$Time <= 180, ]
  eeg <- eeg[, c("Subject", "Item", "Time", "Fz",
    "FreqB", "LengthB", "WMC")]

  # mean center FreqB
  eeg$FreqBc <- eeg$FreqB - mean(eeg$FreqB)
  # split FreqBc into 3 categories. Doesn't make sense,
  # but it's merely for example
```

```
eeg$FreqBdc <- "high"
eeg$FreqBdc[eeg$FreqBc<=quantile(eeg$FreqBc)[3]] <- "mid"
eeg$FreqBdc[eeg$FreqBc<=quantile(eeg$FreqBc)[2]] <- "low"
eeg$FreqBdc <- as.factor(eeg$FreqBdc)
eeg$FreqBdc <- relevel(eeg$FreqBdc, "low")

# mean center LengthB
eeg$LengthBc <- eeg$LengthB - mean(eeg$LengthB)

# mean center WMC
eeg$WMCc <- eeg$WMC - mean(eeg$WMC)

############################################
#      Demonstrate plotDensity3d.fnc.      #
############################################
plotDensity3d.fnc(x = sort(unique(eeg$WMCc)),
  y = sort(unique(eeg$LengthBc)))

############################################
#        Demonstrate plotRaw3d.fnc.        #
############################################
plotRaw3d.fnc(data = eeg, response = "Fz", pred = "WMCc",
  intr = "LengthBc", plot.type = "persp", theta = 150)

############################################
#       Analyze data. Demonstrate model    #
#       selection, and diagnostic plots.   #
#       Also demonstrate forward fitting   #
#       of random effects and back fitting #
#       of fixed effects. Finally,         #
#       demonstrate pamer.fnc.             #
############################################
library(lme4)
# fit initial model
m0 <- lmer(Fz ~ (FreqBdc + LengthBc + WMCc)^2 + (1 | Subject),
  data = eeg)
m1 <- lmer(Fz ~ (FreqBdc + LengthBc + WMCc)^2 + (1 | Subject) +
  (1 | Item), data = eeg)

# which model to choose?
relLik(m0, m1)

# choose m1
# check model assumptions
mcp.fnc(m1)

# remove outliers
eeg <- romr.fnc(m1, eeg, trim = 2.5)
eeg$n.removed
eeg$percent.removed
eeg<-eeg$data

# update model
```

```
m1 <- lmer(Fz ~ (FreqBdc + LengthBc + WMCc)^2 + (1 | Subject) +
  (1 | Item), data = eeg)

# re-check model assumptions
mcp.fnc(m1)

# forward-fit random effect structure (simple for the purposes
# of the example).
m2 <- ffRanefLMER.fnc(model = m1, ran.effects =
  c("(0 + LengthBc | Subject)", "(0 + WMCc | Item)"),
  log.file = FALSE)

# backfit model m2. In this case, could use bfFixefLMER_t.fnc instead.
m3 <- bfFixefLMER_F.fnc(m2, log.file = FALSE)

# The calls to ffRanefLMER.fnc and bfFixefLMER_F.fnc could
# be replaced by a call to fitLMER.fnc. In this latter case, however,
# bfFixefLMER_F.fnc would be called first, then the random effect
# structure would be forward fitted, and finally teh fixed effects
# would be backfitted again.
m3b <- fitLMER.fnc(model = m1, ran.effects = c("(0 + LengthBc | Subject)",
  "(0 + WMCc | Item)"), backfit.on = "F", log.file = FALSE)
pamer.fnc(m3b)
# The results are the same. This may not necessarily be the case
# elsewhere. First forward fitting the random effect structure and
# then backfitting the fixed effects, potentially pruning irrelevant
# random effects, is probably the best approach. Nonetheless, there is
# no hard evidence to this effect.

# check model assumptions
mcp.fnc(m3)

# check significance of model terms
pamer.fnc(m3)

#############################################
#       Demonstrate mcposthoc.fnc and       #
#       summary.mcposthoc.                  #
#############################################
# Only the intercept is significant. For purposes of the
# example, let's perform a posthoc analysis on FreqBdc on
# model m2.
m2.ph <- mcposthoc.fnc(model = m2, var = list(ph1 = "FreqBdc"))

# Now check if and how the different levels differ between
# each other. First check high vs mid and high vs low:
summary(m2.ph, term = "FreqBdchigh")
# Then low vs mid (the low vs high row is redundant from the
# above summary):
summary(m2.ph, term = "FreqBdcmid")
# Note that none of the levels differ from each other. Indeed,
# the backfitting process indicated that the model only has an
# intercept (i.e., the FreqBc factor variable was not significant).
```

```
 # Just to show how one would look at posthocs for interactions. Let's
 # look at the effect of Length at each FreqB bin:
 summary(object = m2.ph, term = "LengthBc")
 # Does Length effect different Freq bins? Start with low
 # versus mid and high
 smry <- summary(object = m2.ph, term = "FreqBdchigh:LengthBc")
 # then mid versus low and high
 smry <- summary(object = m2.ph, term = "FreqBdcmid:LengthBc")

 ############################################
 #       Demonstrate `revived' version of   #
 #       plotLMER.fnc and plotLMER3d.fnc.   #
 ############################################
 # Generate plot for Length X Freq with function plotLMER.fnc.
 plotLMER.fnc(m2, pred = "LengthBc", intr = list("FreqBdc",
   levels(eeg$FreqBdc), "beg", list(1 : 3, 1 : 3)))

 # Plotting the Length:WMC interaction with plotLMER3d.fnc. It'll
 # take a little bit of time.
 plotLMER3d.fnc(m2,"LengthBc","WMCc")
 # Plot it a second time to demonstrate caching. You can notice the
 # speed-up.
 plotLMER3d.fnc(m2,"LengthBc","WMCc")


 ############################################
 #       Demonstrate modeling and           #
 #       backfitting of glmer.              #
 ############################################
 # Split FreqBc into 2 categories.
 eeg$FreqBdc <- "high"
 eeg$FreqBdc[eeg$FreqBc<=median(eeg$FreqBc)] <- "low"
 eeg$FreqBdc <- as.factor(eeg$FreqBdc)
 eeg$FreqBdc <- relevel(eeg$FreqBdc, "low")

 # Fit glmer model.
 m4 <- glmer(FreqBdc ~ (Fz + LengthBc + WMCc)^2 + (1 | Subject),
family = "binomial", data = eeg)
 summary(m4)
 pamer.fnc(m4)

 # Back fit fixed effects, forward fit random effects, and then
 # re-back fit fixed effects. Need to set argument backfit.on to "t".
 m5 <- fitLMER.fnc(model = m4, ran.effects = "(0 + LengthBc | Subject)",
backfit.on = "t", log.file = FALSE)
 summary(m5)
 pamer.fnc(m5)

 # Plot the 2-way interaction.
 plotLMER.fnc(m5, pred = "Fz", intr = list("LengthBc",
quantile(eeg$LengthBc), "med",list(1:5,1:5)))
```

```
 # Look at the same plot, but in 3d.
 plotLMER3d.fnc(m5, pred = "Fz", intr = "LengthBc")

 ###########################################
 #       Test backfitting on AIC,          #
 #       BIC, llrt, relLik.AIC, and        #
 #       relLik.BIC.                       #
 ###########################################
 # AIC
 m.test <- bfFixefLMER_F.fnc(m2, method = "AIC",
log.file = FALSE)
 m.test <- bfFixefLMER_t.fnc(m2, method = "AIC",
log.file = FALSE)
 m.test <- bfFixefLMER_t.fnc(m4, method = "AIC",
log.file = FALSE)
 m.test <- bfFixefLMER_F.fnc(m4, method = "AIC",
log.file = FALSE)

 # BIC
 m.test <- bfFixefLMER_F.fnc(m2, method = "BIC",
log.file = FALSE)
 m.test <- bfFixefLMER_t.fnc(m2, method = "BIC",
log.file = FALSE)
 m.test <- bfFixefLMER_t.fnc(m4, method = "BIC",
log.file = FALSE)

 # llrt
 m.test <- bfFixefLMER_F.fnc(m2, method = "llrt",
log.file = FALSE)
 m.test <- bfFixefLMER_t.fnc(m2, method = "llrt",
log.file = FALSE)
 m.test <- bfFixefLMER_t.fnc(m4, method = "llrt",
log.file = FALSE)

 # relLik.AIC
 m.test <- bfFixefLMER_F.fnc(m2, method = "relLik.AIC",
log.file = FALSE)
 m.test <- bfFixefLMER_t.fnc(m2, method = "relLik.AIC",
log.file = FALSE)
 m.test <- bfFixefLMER_t.fnc(m4, method = "relLik.AIC",
log.file = FALSE)

 # relLik.BIC
 m.test <- bfFixefLMER_F.fnc(m2, method = "relLik.BIC",
log.file = FALSE)
 m.test <- bfFixefLMER_t.fnc(m2, method = "relLik.BIC",
log.file = FALSE)
 m.test <- bfFixefLMER_t.fnc(m4, method = "relLik.BIC",
log.file = FALSE)

## End(Not run)
```

`bfFixefLMER_F.fnc` *Back-fits an LMER model on p-values from ANOVA, llrt, AIC, BIC, relLik.AIC or relLik.BIC.*

Description

This function back-fits an initial LMER model either on upper- or lower-bound *p*-values obtained from function `pamer.fnc`, log-likelihood ratio testing (LLRT), AIC, BIC, relLik.AIC, or relLik.BIC. Note that this function CANNOT be used with generalized linear mixed-effects models (`glmer`s).

Usage

```
bfFixefLMER_F.fnc(model, item = FALSE,
method = c("F", "llrt", "AIC", "BIC", "relLik.AIC",
"relLik.BIC"), threshold = NULL, alpha = NULL,
alphaitem = NULL, prune.ranefs = TRUE,
p.value = "upper", set.REML.FALSE = TRUE,
keep.single.factors=FALSE, reset.REML.TRUE = TRUE,
log.file = NULL)
```

Arguments

`model`	A `mer` object (fitted by function `lmer`). Note that this function cannot be used with generalized linear mixed-effects models (`glmer`s).
`item`	Whether or not to evaluate the addition of by-item random intercepts to the model, evaluated by way of log-likelihood ratio test. Either `FALSE` (the default) or the column name (quoted) of the item identifier (e.g., `"Item"`, or `"Word"`).
`method`	Backfitting method. One of "F" (*p*-value), "llrt", "AIC", "BIC", "relLik.AIC", or "relLik.BIC" (relative likelihood, see function `relLik`). Defaults to `F`. You can find information regarding differences between AIC and BIC from `http://methodology.psu.edu/eres`
`threshold`	Method-specific threshold for parameter selection. It refers to alpha in the case of "F" and "llrt", to the minimum reduction in likelihood in the case of "AIC" and "BIC", or to the minimum difference in probability in the case of "relLik.AIC" and "relLik.BIC". Defaults `NULL`, which means `0.05` for "F" and "llrt", 5 for "AIC" and "BIC", and 4 for "relLik.AIC" and "relLik.BIC".
`alpha`	If the method is `F`, it is the *p*-value (from `pamer.fnc`) above which a model term is dropped. In this case, it defaults to the value passed to argument `threshold`, i.e., 0.05. Otherwise it is the *p*-value threshold above which a test (see `method`) is performed between a model with the term under consideration and a simpler model without it (in this case, defaults to `0`, i.e. all terms will be tested).
`alphaitem`	Alpha value for the evaluation of by-item random intercepts. Defaults to `0.05` or to the specified threshold.
`prune.ranefs`	Logical. Whether to remove any random effect for which its variable is not also present in the fixed effects structure (with the exception of the grouping variables such as `"Subjects"` and `"Items"`). Defaults to `TRUE`. For example, if

	the random effects structure contains the terms `Condition + ROI + Group`, and the random effects structure contains the terms `(1 | Subject) + (0 + TrialNum | Subject)`, the random effect `(0 + TrialNum | Subject)` will be pruned from the model given that it is not in the model's fixed effects structure.
`p.value`	If `method = "F"`, whether to use upper-bound ("upper"; the default) or lower-bound ("lower") *p*-values during backfitting.
`set.REML.FALSE`	Logical. Whether or not to set `REML` to `FALSE`. Defaults to `TRUE`.
`keep.single.factors`	Logical. Whether or not main effects are kept (not subjected to testing and reduction). Defaults to `FALSE`.
`reset.REML.TRUE`	Logical. Whether or not to re-set the back-fitted model to `REML = TRUE`.
`log.file`	Whether a back-fitting log should be saved. Defaults to `NULL`, which means that a log is saved in a temporary folder with the file name `file.path(tempdir(), paste("bfFixefLMER_F_log_", gsub(":", "-", gsub(" ", "_", date())), ".txt", sep = ""))`. The path and file name of the log can be changed to whatever the use wishes. Set to `FALSE` to disable.

Details

The back-fitting process works as follows:

1. If argument `method` is not set to `F`, `REML` is set to `FALSE`;
2. First consider only highest-order interaction model terms:
    (a) If `method` is `F`, the model term with the highest ANOVA *p*-value is identified. If this *p*-value is higher than `alpha`,the model term is removed and a new model is fitted. This is repeated for each model term that has a *p*-value higher than the `alpha` value. The algorithm then moves on to step (b). If `method` is not `F`, the model term with the lowest *p*-value is identified and the following is evaluated:
        i. A new model without this model term is fitted;
        ii. The more complex and simpler models are compared by way of a log-likelihood ratio test in case `method` is "llrt", by way of AIC or BIC values in case `method` is "AIC" or "BIC", or by calculating the `relLik` based on AIC or BIC in case `method` is "relLik.AIC" or "relLik.BIC". If the result determines that the term under consideration does not increase model fit, it is removed; otherwise it is kept.
        iii. Move on to the next model term with the smallest *p*-value smaller than `alpha` and repeat steps (i)–(iii).
    (b) Once all highest-order interaction terms have been evaluated, go down to the second highest order interactions: Repeat steps (ai)–(aiii) with the following addition: If a term would be removed from the model, but it is part of a high-order interaction, keep it. Once all terms of the interaction level have been evaluated, move down to the next lower-order level until main effects have been evaluated, after which the process stops. If `keep.single factors = TRUE`, the process stops after the evaluation of all interaction terms.
3. If argument `method` is set to something else other than "F", set `reset.REML.TRUE` to `TRUE` (default) unless otherwise specified.

In brief, if `method` is set to "F", a term remains in the model if its *p*-value is equal to or greater than `alpha`; if `method` is set to something else, a term remains in the model if

1. its *p*-value from the ANOVA is equal to or smaller than `alpha`;
2. it significantly increases model fit as determined by the specified method;
3. it is part of a significant higher-order interaction term.

This backfitting method was used in Newman, Tremblay, Nichols, Neville, and Ullman (2012). If factorial terms are included in the initial model, back-fitting on F is recommended.

Value

A `mer` model with back-fitted fixed effects is returned and a log of the back-fitting process is printed on screen and (by default) in a log file in a temporary file.

Warnings

Upper-bound *p*-values can be anti-conservative, while lower-bound *p*-values can be conservative. See function `pamer.fnc`.

Note

If you get this error:

```
Error in model.frame.default(data = ..2, formula = log_Segment_Duration ~  :
The ... list does not contain 2 elements
```

It is probably because you updated the model using function `update` and the data now appears as `data = ..2` or something similar to this. You can check this by typing `model@call`. If this is the case, re-fit your model as `lmer(DV ~ IV + IV + (RANEF), data = dat)`.

Author(s)

Antoine Tremblay, Statistics Canada, `<trea26@gmail.com>` and Johannes Ransijn `<johannesransijn@gmail.com>`.

References

Newman, A.J., Tremblay, A., Nichols, E.S., Neville, H.J., and Ullman, M.T. (2012). The Influence of Language Proficiency on Lexical Semantic Processing in Native and Late Learners of English. *Journal of Cognitive Neuroscience*, *25*, 1205–1223.

See Also

`bfFixefLMER_t.fnc`; `ffRanefLMER.fnc`; `fitLMER.fnc`; `mcposthoc.fnc`; `pamer.fnc`; `mcp.fnc`; `relLik`; `romr.fnc`; `perSubjectTrim.fnc`.

Examples

```
# see example in LMERConvenienceFunctions help page.
```

`bfFixefLMER_t.fnc` *Back-fits an LMER model on absolute t-value and, optionally, on LLRT.*

Description

This function back-fits an initial LMER model on *t*-values, and, if enabled, log-likelihood ratio testing. Note that, this function CAN be used with generalized linear mixed-effects models (`glmers`).

Usage

```
bfFixefLMER_t.fnc(model, item = FALSE,
method = c("t", "z", "llrt", "AIC", "BIC", "relLik.AIC",
"relLik.BIC"), threshold = NULL, t.threshold = NULL,
alphaitem = NULL, prune.ranefs = TRUE, set.REML.FALSE = TRUE,
keep.single.factors=FALSE, reset.REML.TRUE = TRUE,
log.file = NULL)
```

Arguments

`model`	A `mer` object (fitted by function `lmer`). Note that this function can be used with generalized linear mixed-effects models (`glmers`).
`item`	Whether or not to evaluate the addition of by-item random intercepts to the model, evaluated by way of log-likelihood ratio test. Either `FALSE` (the default) or the column name (quoted) of the item identifier (e.g., `"Item"`, or `"Word"`).
`method`	Backfitting method. One of "t" (lmer), "z" (glmer), "llrt", "AIC", "BIC", "relLik.AIC", or "relLik.BIC" (the latter two are based on relative likelihood, see function `relLik`). Defaults to "t". You can find information regarding differences between AIC and BIC from `http://methodology.psu.edu/eresources/ask/sp07`.
`threshold`	Method-specific threshold for parameter selection. It refers to the minimum *t*/*z*-value in the case of "t" or "z", to the alpha value in the case of "llrt", to the minimum reduction in likelihood in the case of "AIC" and "BIC", or to the minimum difference in probability in the case of "relLik.AIC" and "relLik.BIC". Defaults `NULL`, which means 2 for "t" and "z", `0.05` for "llrt", 5 for "AIC" and "BIC", and 4 for "relLik.AIC" and "relLik.BIC".
`t.threshold`	Defaults to `NULL`. If the `method` = `"t"` or `method` = `"z"`, it is the *t*/*z*-value below which a model term is dropped (if `t.threshold` = `NULL`, it will be set to 2). Otherwise it is the threshold for *t*/*z*-value below which a test (see `method`) is performed between a model with the term under consideration and a simpler model without it (if `t.threshold` = `NULL`, it is set to `Inf`, which means that all terms are tested.
`alphaitem`	Alpha value for the evaluation of by-item random intercepts. Defaults to `0.05` or to the specified threshold in case `method` is `llrt`.
`prune.ranefs`	Logical. Whether to remove any random effect for which its variable is not also present in the fixed effects structure (with the exception of the grouping

	variables such as `"Subjects"` and `"Items"`). Defaults to `TRUE`. For example, if the random effects structure contains the terms `Condition + ROI + Group`, and the random effects structure contains the terms `(1 | Subject) + (0 + TrialNum | Subject)`, the random effect `(0 + TrialNum | Subject)` will be pruned from the model given that it is not in the model's fixed effects structure.
`set.REML.FALSE`	Logical. Whether or not to set REML to `FALSE`. Defaults to `TRUE`. Not used for glmer models.
`reset.REML.TRUE`	Logical. Whether or not to re-set the back-fitted model to `REML = TRUE`. Not used for glmer models.
`keep.single.factors`	Logical. Whether or not main effects are kept (not subjected to testing and reduction). Defaults to `FALSE`.
`log.file`	Whether a back-fitting log should be saved. Defaults to `NULL`, which means that a log is saved in a temporary folder with the file name `file.path(tempdir(), paste("bfFixefLMER_F_log_", gsub(":", "-", gsub(" ", "_", date())), ".txt", sep = ""))`. The path and file name of the log can be changed to whatever the use wishes. Set to `FALSE` to disable.

Details

The back-fitting process works as follows:

1. If argument `method` is not set to "t", `REML` is set to `FALSE`;
2. First consider only highest-order interaction model terms:
   (a) If `method` is "t" or "z", the model term with the lowest *t*/*z*-value is identified. If this *t*/*z*-value is smaller than `threshold`, the model term is removed and a new model is fitted. This is repeated for each model term for term that has a *t*-value smaller than the threshold value. The algorithm then moves on to step (b). If `method` is not "t" or "z", the model term with the lowest *t*/*z*-value-value is identified and the following is evaluated:
      i. A new model without this model term is fitted;
      ii. The more complex and simpler models are compared by way of a log-likelihood ratio test in case `method` is "llrt", by way of AIC or BIC comparison if `method` is "AIC" "BIC", or by calculating the `relLik` based on AIC or BIC in case `method` is "relLik.AIC" or "relLik.BIC". If the result determines that the term under consideration does not increase model fit, it is removed; otherwise it is kept.
      iii. Move on to the next model term with the smallest *t*/*z*-value smaller than `threshold` and repeat steps (i)–(iii).
   (b) Once all highest-order interaction terms have been evaluated, go down to the second highest order interactions: Repeat steps (ai)–(aiii) with the following addition: If a term would be removed from the model, but it is part of a high-order interaction, keep it. Once all terms of the interaction level have been evaluated, move down to the next lower-order level until main effects have been evaluated, after which the process stops. If `keep.single factors = TRUE`, the process stops after the evaluation of all interaction terms.
3. If argument `method` is set to something other than `t` or `z`, set `reset.REML.TRUE` to `TRUE` (default) unless otherwise specified.

In brief, if `method` is set to "t" or "z", a term remains in the model if its *t*/*z*-value is equal to or greater than `threshold`; if `method` is set to something else, a term remains in the model if

1. its *t*/*z*-value is equal to or greater than `threshold`;
2. it significantly increases model fit as determined by the specified method;
3. it is part of a significant interaction term.

This backfitting method was used in Tremblay & Tucker (2011). If factorial terms with more than two levels are included in the initial model, back-fitting on F is recommended.

Value

A `mer` model with back-fitted fixed effects (on `t`-values) is returned and a log of the back-fitting process is printed on screen and (by default) in a log file.

Note

If you get this error:

```
Error in model.frame.default(data = ..2, formula = log_Segment_Duration ~  :
  The ... list does not contain 2 elements
```

It is probably because you updated the model using function `update` and the data now appears as `data = ..2` or something similar to this. You can check this by typing `model@call`. If this is the case, re-fit your model as `lmer(DV ~ IV + IV + (RANEF), data = dat)`.

Author(s)

Antoine Tremblay, Statistics Canada, <`trea26@gmail.com`> and Johannes Ransijn <`johannesransijn@gmail.com`>.

References

Tremblay, A. and Tucker B. V. (2011). The Effects of N-gram Probabilistic Measures on the Processing and Production of Four-word Sequences. *The Mental Lexicon*, *6(2)*, 302–324.

See Also

`bfFixefLMER_F.fnc`; `ffRanefLMER.fnc`; `fitLMER.fnc`; `mcposthoc.fnc`; `pamer.fnc`; `mcp.fnc`; `relLik`; `romr.fnc`; `perSubjectTrim.fnc`.

Examples

```
# see example in LMERConvenienceFunctions help page.
```

`cd` *Change directory.*

Description

Change directory to the one corresponding to the row number listed by function `f`.

Usage

```
cd(dir)
```

Arguments

`dir`	The row number corresponding to the directory list returned by function `f`.

Value

Change directory to the selected one.

Author(s)

Antoine Tremblay, Statistics Canada, <`trea26@gmail.com`>

See Also

`f`; `cdf`; `cdup`; `setwd`

`cdf` *Change directory; list files and directories in new directory using function `f`.*

Description

Change directory to the one corresponding to the row number returned by function `f`.

Usage

```
cdf(dir)
```

Arguments

`dir`	The row number corresponding to the directory listed by function `f`.

Value

Cheange to new directory and list files and directories in new directory using function `f`.

Author(s)

Antoine Tremblay, Statistics Canada, <`trea26@gmail.com`>

See Also

`f`; `cd`; `cdup`; `setwd`

`cdup` *Change directory one level up.*

Description

Change directory one level up and list directory and files in new directory.

Usage

```
cdup()
```

Value

Change directory one level up.

Author(s)

Antoine Tremblay, Statistics Canada, <trea26@gmail.com>

See Also

`f`; `cd`; `cdf`; `setwd`

`cn` *List the column names of a data frame in matrix format.*

Description

The colum names of the specified data frame are listed in matrix format, that is, each one appears in one row preceded by the row number.

Usage

```
cn(data.frame)
```

Arguments

`data.frame`	A data frame.

Value

A matrix containing the column names of the data frame.

Author(s)

Antoine Tremblay, Statistics Canada, `<trea26@gmail.com>`

See Also

`colnames`

`f` *List files and directories in current directory.*

Description

List files and directories in current directory in matrix format. Each row is preceded by a row number.

Usage

```
f(path = ".", pattern = NULL, all.files = FALSE,
full.names = FALSE, recursive = FALSE, ignore.case = FALSE)
```

Arguments

`path`	A character vector of full path names; the default corresponds to the working directory `getwd()`. Missing values will be ignored.
`pattern`	An optional regular expression. Only file names which match the regular expression will be returned.
`all.files`	Logical. If `FALSE`, only the names of visible files are returned. If `TRUE`, all file names will be returned.
`full.names`	Logical. If `TRUE`, the directory path is prepended to the file names. If `FALSE`, only the file names are returned.
`recursive`	Logical. Should the listing recurse into directories?
`ignore.case`	Logical. Should pattern-matching be case-insensitive?

Value

A matrix containing the names of the files and directories, preceded by a row number, in the specified directories. If a path does not exist or is not a directory or is unreadable it is skipped, with a warning.

The files are sorted in alphabetical order, on the full path if `full.names = TRUE`. Directories are included only if `recursive = FALSE`.

Note

File naming conventions are platform dependent. `recursive = TRUE` is not supported on all platforms and may be ignored (with a warning).

Author(s)

Antoine Tremblay, Statistics Canada, `<trea26@gmail.com>`

See Also

`list.files`

Examples

```
f()
```

`ffRanefLMER.fnc` *Forward-fit the random effect structure of an LMER model.*

Description

Forward-fit an LMER model's random effect structure by comparing a model without one of the specified random effects and a model with it by way of log-likelihood ratio testing. If the more complex model is a significantly better fit, the random effect is kept, otherwise it is dropped. This function can now be used with generalized linear mixed-effects models (`glmers`).

Usage

```
ffRanefLMER.fnc(model, ran.effects = list(ran.intercepts =
as.character(), slopes = as.character(), corr = as.character(),
by.vars = as.character()), alpha = 0.05, if.warn.not.add = TRUE,
log.file = NULL)
```

Arguments

`model`	A `mer` object (fitted by function `lmer`). This function can now be used with generalized linear mixed-effects models (`glmers`).
`ran.effects`	Can be either a vector or a list. In the former case, the random effects to be evaluated are provided. For example `c("(1 + Frequency | Subject)", "(0 + Length | Subject)", "(1 + NSynSet | Subject)")`. In the latter case, the list can be composed of (i) a vector of random intercepts to be evaluated (`ran.intercepts`), (ii) a vector of random slopes to be evaluated (`slopes`), (iii) a vector specifying, for each element of `slopes`, whether the correlation between the slope and by-variables specified in `by.vars` should be added (`corr`), and (iv) a vector of “by” variables for the random slopes (`by.vars`). Values that can be supplied to the `corr` argument are `1` (add correlation), `0` (do not add correlation), and `NA` (for when the "slope" is a factor variable). Note that if a term in `slopes` is a factor

variable, the `corr` value tied to it will be automatically set to `NA`. Also note that if no values are supplied to `corr`, a vector of 0 as long as the `slopes` vector will be automatically supplied. For example `list(ran.intercepts = "Word", slopes = c("Frequency", "Length", "NSynSet","Class"), corr = c(0, 0, 1, NA), by.vars = "Subject")`. Another example is `list(slopes = c("Trial", "Class"), by.vars = "Subject")`, where the `corr` argument will be equal to `c(0, NA)`.

`alpha`	Level of significance for log-likelihood ratio test. Defaults to 0.05.
`if.warn.not.add`	Logical. If a warning is issued after fitting a model with a new random effect (e.g., `false` convergence or the like), should the random effect nevertheless be evaluated? Defaults to `TRUE`, meaning that if such a warning is issued, the random effect will not be added to the random effects structure of the model. If set to `FALSE`, the random effect will be evaluated for inclusion as any other random effects would be via log likelihood ratio testing even if a warning is issued.
`log.file`	Should the back-fitting log be saved? Defaults to `NULL`, which means that a log file is saved in a temporary folder as `paste("ffRanefLMER_log_", gsub(":", "-", gsub(" ", "_", date())), ".txt", sep = "")`. The path and file name of the log can be changed to whatever the use wishes. Set to `FALSE` to disable.

Value

A `mer` object with forward-fitted random effect structure as well as a log of the process is printed on screen and, optionally, printed in a log file.

Note

The removal of a random effect from the random effects structure if the variables that compose it are not also in the fixed effects structure has been turned off in this version.

Author(s)

Antoine Tremblay, Statistics Canada, `trea26@gmail.com`.

References

Pinheiro, J.C. and Bates, D.M. (2000). *Mixed Effects Models in S and S-Plus*. New York: Springer.

See Also

`bfFixefLMER_F.fnc`; `bfFixefLMER_t.fnc`; `fitLMER.fnc`; `mcposthoc.fnc`; `pamer.fnc`; `mcp.fnc`; `romr.fnc`; `perSubjectTrim.fnc`.

Examples

```
# see example in LMERConvenienceFunctions help page.
```

`fitLMER.fnc` *Back-fit fixed effects and forward-fit random effects of an LMER model.*

Description

The function follows these steps: (1) If `llrt` is set to `TRUE`, set `REML` to `FALSE` (unless specified otherwise); (2) back-fit initial model either on F- (by default) or on t/z-values; (3) forward-fit random effects; (4) re-back-fit fixed effects; (5) if `llrt` is set to `TRUE`, set `REML` to `TRUE` (unless specified otherwise). Note that, this function CAN be used with generalized linear mixed-effects models (`glmers`).

Usage

```
fitLMER.fnc(model, item = FALSE, backfit.on = c("F",
"t"), method = c("F", "t", "z", "llrt", "AIC", "BIC", "relLik.AIC",
"relLik.BIC"), threshold = NULL, t.threshold = NULL,
ran.effects = list(ran.intercepts = as.character(),
slopes = as.character(), corr = as.character(),
by.vars = as.character()), alpha = NULL, alphaitem = NULL,
if.warn.not.add = TRUE, prune.ranefs = TRUE, p.value = "upper",
set.REML.FALSE = TRUE, keep.single.factors = FALSE,
reset.REML.TRUE = TRUE, log.file.name = NULL)
```

Arguments

`model`	A `mer` object (fitted by function `lmer`). This function can be used with generalized linear mixed-effects models (`glmers`) if argument `backfit.on` is set to "t", but not if it is set to "F".
`item`	Whether or not to evaluate the addition of by-item random intercepts to the model, evaluated by way of log-likelihood ratio test. Either `FALSE` (the default, does not evaluate this addition) or the column name (quoted) of the item identifier (e.g., `"Item"`, `"Word"`).
`backfit.on`	Either "F" (default) or "t". Refers to the statistic which will be used to determine which term to test and potentially remove from the model. If you are backfitting a generalized linear mixed-effects model (`glmer`), make sure to set `backfit.on` to "t"; the algorithm efectively backfits on "z".
`method`	Backfitting method. One of "F" (p-value), "t" (t statistic), "z" (z statistic), "llrt", "AIC", "BIC", "relLik.AIC", or "relLik.BIC" (the latter two are based on relative likelihood, see function `relLik`). Defaults to "t". You can find information regarding differences between AIC and BIC from `http://methodology.psu.edu/eresources/ask/sp07`
`threshold`	Method-specific threshold for parameter selection. It refers to alpha in the case of "F" and "llrt", to the t/z-value in case of "t" or "z", to the minimum reduction in likelihood in the case of "AIC" and "BIC", or to the minimum difference in probability in the case of "relLik.AIC" and "relLik.BIC". Defaults `NULL`, which means `0.05` for "F" and "llrt", 2 for "t", 5 for "AIC" and "BIC", and 4 for "relLik.AIC" and "relLik.BIC".

`t.threshold`	Defaults to `NULL`. If the `method = "t"` or `method = "z"`, it is the *t*/*z*-value below which a model term is dropped (if `t.threshold = NULL`, it will be set to 2). Otherwise it is the threshold for *t*/*z*-value below which a test (see `method`) is performed between a model with the term under consideration and a simpler model without it (if `t.threshold = NULL`, it is set to `Inf`, which means that all terms are tested.
`ran.effects`	Can be either a vector or a list. In the former case, the random effects to be evaluated are provided. For example `c("(1 + Frequency | Subject)", "(0 + Length | Subject)", "(1 + NSynSet | Subject)")`. In the latter case, the list can be composed of (i) a vector of random intercepts to be evaluated (`ran.intercepts`), (ii) a vector of random slopes to be evaluated (`slopes`), (iii) a vector specifying, for each element of `slopes`, whether the correlation between the slope and by-variables specified in `by.vars` should be added (`corr`), and (iv) a vector of "by" variables for the random slopes (`by.vars`). Values that can be supplied to the `corr` argument are `1` (add correlation), `0` (do not add correlation), and `NA` (for when the "slope" is a factor variable). Note that if a term in `slopes` is a factor variable, the `corr` value tied to it will be automatically set to `NA`. Also note that if no values are supplied to `corr`, a vector of 0 as long as the `slopes` vector will be automatically supplied. For example `list(ran.intercepts = "Word", slopes = c("Frequency", "Length", "NSynSet","Class"), corr = c(0, 0, 1, NA), by.vars = "Subject")`. Another example is `list(slopes = c("Trial", "Class"), by.vars = "Subject")`, where the `corr` argument will be equal to `c(0, NA)`.
`alpha`	If the method is `F`, it is the *p*-value (from `pamer.fnc`) above which a model term is dropped. In this case, it defaults to the value passed to argument `threshold`, i.e., 0.05. Otherwise it is the *p*-value threshold above which a test (see `method`) is performed between a model with the term under consideration and a simpler model without it (in this case, defaults to `0`, i.e. all terms will be tested).
`alphaitem`	Alpha value for the evaluation of by-item random intercepts. Defaults to `0.05` or to the specified threshold.
`if.warn.not.add`	Logical. If a warning is issued after fitting a model with a new random effect (e.g., `false convergence` or the like), should the random effect nevertheless be evaluated? Defaults to `TRUE`, meaning that if such a warning is issued, the random effect will not be added to the random effects structure of the model. If set to `FALSE`, the random effect will be evaluated for inclusion as any other random effects would be via log likelihood ratio testing even if a warning is issued.
`prune.ranefs`	Logical. Whether to remove any random effect for which its variable is not also present in the fixed effects structure (with the exception of the grouping variables such as `"Subjects"` and `"Items"`). Defaults to `TRUE`. For example, if the random effects structure contains the terms `Condition + ROI + Group`, and the random effects structure contains the terms `(1 | Subject) + (0 + TrialNum | Subject)`, the ranedom effect `(0 + TrialNum | Subject)` will be pruned from the model given that it is not in the model's fixed effects structure.
`p.value`	Whether to use upper-bound ("upper"; the default) or lower-bound ("lower") *p*-values when back-fitting with method "F".

`set.REML.FALSE`	Logical. Whether or not to set `REML` to `FALSE`. Defaults to `FALSE`.
`reset.REML.TRUE`	Logical. Whether or not to re-set the back-fitted model to `REML = TRUE`.
`keep.single.factors`	Logical. Whether or not main effects are kept (not subjected to testing and reduction). Defaults to `FALSE`.
`log.file.name`	Should the back-fitting log be saved? Defaults to `NULL`, which means that a log file is saved in a temporary folder (platform dependent) as `file.path(tempdir(), paste("fitLMER_log_", gsub(":", "-", gsub(" ", "_", date())), ".txt", sep = ""))`. The path and file name of the log can be changed to whatever the use wishes. Set to `FALSE` to disable.

Details

The process has three stages. In the first stage, either `bfFixefLMER_F.fnc` or `bfFixefLMER_t.fnc` is called (depending on the user's choice) and the fixed effects are back-fitted accordingly. In the second stage, `ffRanefLMER.fnc` is called and random effects are forward-fitted. In the third stage, the fixed effects are back-fitted again. This is done because the inclusion of certain random effects sometimes renders certain fixed effects non-significant. This process was used in Tremblay and Tucker (2011) and in Newman, Tremblay, Nichols, Neville, and Ullman (2012).

If, for example, you have many analyses to run and a cluster is available, write a bash script that will create (1) `.R` files that will relevel the conditions and update the model, and (2) an associated `.sh` job submission script to submit the `.R` files. For example, let's consider two ERP analyses all in a time window ranging from 100 to 250 ms. Two three-way interactions were considered: Position (factor; 1 to 6) X Length of the second word of a four-word sequence (e.g., *in the middle of*) X Working Memory Capacity score (continuous, from 0 to 100) and Trial (continuous; 1 to 432) X Length X Working Memory Capacity. Analyses were performed at electrodes Fp1 Fp2 AF3 AF4 F7 F3 Fz F4 F8 FC5 FC1 FC2 FC6 T7 C3 Cz C4 T8 CP5 CP1 CP2 CP6. See Tremblay and Newman (In preparation) for more details. The analysis script named `Fp1-CP6_100250.sh` we used on the ACEnet cluster is as follows:

```
electrodes=(Fp1 Fp2 AF3 AF4 F7 F3 Fz F4 F8 FC5 FC1 FC2 FC6 T7 C3 Cz C4 T8 CP5 CP1 CP2 CP6)
 for e in ${electrodes[*]}; do
    export E=$e;
    # create .R script to load data, perform necessary manipulations
    # and perform the analysis using fitLMER.fnc
    echo 'e<-Sys.getenv("E")' > $e".R"
    echo 'load("../data/eeg600_trim_v2.rda")' >> $e".R"
    echo 'dat0<-dat' >> $e".R"
    echo 'rm(dat);gc(T,T)' >> $e".R"
  echo 'dat <- dat0[dat0$Time >= 100 & dat0$Time <= 250, , drop = TRUE]' >> $e".R"
    echo 'dat <- dat[dat$Electrode == e, , drop = TRUE]' >> $e".R"
    echo 'subj<-sort(unique(dat$Subject))' >> $e".R"
    echo 'for(i in subj){' >> $e".R"
    echo 'tmp<-dat[dat$Subject==i,,drop=TRUE]' >> $e".R"
    echo 'tmp$newfact<-paste(tmp$Block,tmp$Position,sep="_")' >> $e".R"
    echo 'newvec<-vector("numeric")' >> $e".R"
    echo 'for(j in 1:length(unique(tmp$newfact))){' >> $e".R"
```

```
 echo 'newvec<-c(newvec,rep(j,nrow(tmp[tmp$newfact==unique(tmp$newfact)[j],])))' >> $e”.R”
   echo '}' >> $e”.R”
   echo 'tmp$Trial<-newvec' >> $e”.R”
   echo 'if(grep(i,subj)[1]==1){' >> $e”.R”
   echo 'newdat<-tmp' >> $e”.R”
   echo '}else{' >> $e”.R”
   echo 'newdat<-rbind(newdat,tmp)' >> $e”.R”
   echo '}' >> $e”.R”
   echo '}' >> $e”.R”
   echo 'dat<-newdat' >> $e”.R”
   echo 'dat$Position<-as.factor(dat$Position)' >> $e”.R”
 echo 'm7 <- lmer(Amplitude ~ (Position + Trial)*(LengthBc * WMCc) + ' >> $e”.R”
   echo '(1 | Subject), data = dat)' >> $e”.R”
 echo 'm7b<-fitLMER.fnc(m7,item=”Item”,ran.effects=c(”(0+Trial|Subject)”,' >> $e”.R”
 echo '”(0+LengthBc|Subject)”,”(0+Trial|Item)”,”(0+WMCc|Item)”,' >> $e”.R”
   echo '”(Position|Subject)”))' >> $e”.R”
   echo 'smry<-pamer.fnc(m7b)' >> $e”.R”
 echo 'save(m7b,file=file.path(”..”,”models”,paste(”m7b_”,e,”_100250.rda”,sep=””)))' >> $e”.R
 echo 'save(smry,file=file.path(”..”,”summaries”,paste(”smry_m7b_”,e,”_100250.rda”,sep=””)))'

   ### create the job submission script for the .R file created above
   echo '#$ -S /bin/bash' > ”job.”$e”.sh”
   echo '#$ -cwd' >> ”job.”$e”.sh”
   echo '#$ -j y' >> ”job.”$e”.sh”
   echo '#$ -l h_rt=48:00:00' >> ”job.”$e”.sh”
   echo '#$ -l h_vmem=8G' >> ”job.”$e”.sh”
   echo '#$ -R y' >> ”job.”$e”.sh”
   echo '#$ -N '$e >> ”job.”$e”.sh”
   echo 'R -q -f '$e'.R' >> ”job.”$e”.sh”

   ### submit the job
   qsub  ”job.”$e”.sh”
done;
```

and then type in the console

```
. Fp1-CP6_100250.sh
```

On the ACEnet cluster, this results in 22 independent analyses, simultaneously using a total of 22 cores and 176 GB of RAM. This analysis completes in about 30 minutes to 1 hour.

Value

A `mer` object with back-fitted fixed effects and forward-fitted random effects, as well as a log of the process, which is printed on screen and, optionally, printed in a log file.

Warnings

Upper-bound *p*-values can be anti-conservative, while lower-bound *p*-values can be conservative. See function `pamer.fnc`.

Note

The removal of a random effect from the random effects structure if the variables that compose it are not also in the fixed effects structure has been turned off in this version.

Author(s)

Antoine Tremblay, Statistics Canada, <`trea26@gmail.com`>

References

Baayen, R.H., Davidson, D.J. and Bates, D.M. (2008). Mixed-effects modeling with crossed random effects for subjects and items. Journal of Memory and Language, 59, 390–412.

Newman, A.J., Tremblay, A., Nichols, E.S., Neville, H.J., and Ullman, M.T. (2012). The Influence of Language Proficiency on Lexical Semantic Processing in Native and Late Learners of English. *Journal of Cognitive Neuroscience*, *25*, 1205–1223.

Pinheiro, J.C. and Bates, D.M. (2000). *Mixed Effects Models in S and S-Plus*. New York: Springer.

Tremblay, A. and Tucker B. V. (2011). The Effects of N-gram Probabilistic Measures on the Processing and Production of Four-word Sequences. *The Mental Lexicon*, *6(2)*, 302–324.

See Also

`bfFixefLMER_F.fnc`; `bfFixefLMER_t.fnc`; `ffRanefLMER.fnc`; `mcposthoc.fnc`; `pamer.fnc`; `mcp.fnc`; `relLik`; `romr.fnc`; `perSubjectTrim.fnc`.

Examples

```
# see example LMERConvenienceFunctions help page.
```

`mcp.fnc` *Model criticism plots.*

Description

A function to graph criticism plots for an LMER model (as in Baayen, 2008, chapter 7). Note that this function cannot be used with generalized linear mixed-effects models (GLMERs). Also note that the fourth plot (dffits) is omitted until we can figure out how to calculate dffits for a `merMod` object.

Usage

```
mcp.fnc(model, trim = 2.5, col = "red")
```

Arguments

`model`	A `mer` object (fitted by function `lmer`). Note that, at the moment, this function cannot be used with generalized linear mixed-effects models (GLMERs).
`trim`	Used to plot lines in the fitted ~ standardized residuals plot. The lines correspond to the threshold at which residuals would be or were removed. Defaults to 2.5 (standard deviations above and below the residuals mean).
`col`	Color of the lines added to the quantile-quantile plot and fitted ~ standardized residuals plot. Defaults to red.

Details

The first of the four plots graphs the density of the model residuals. The second plot graphs the quantile-quantile plot (actual standardized residuals versus theoretical quantiles). The third plot illustrates the fitted values versus the standardized residuals. The fourth graph plots the absolute values of the `dffits` of the residuals (not producing this plot as of version 2.2; might come back in future versions).

Value

Returns the four plots described above.

Author(s)

Antoine Tremblay, Statistics Canada, <`trea26@gmail.com`>.

References

Baayen, R.H. (2008). *Analyzing Linguistic Data. A Practical Introduction to Statistics Using R*. Cambridge, UK: Cambridge University Press.

Examples

```
# see example LMERConvenienceFunctions help page.
```

`mcposthoc.fnc` *Posthoc analyses for LMER models using parallel capabilities.*

Description

This function uses the `parallel` package. For each factor level, a slave process is sent to one of the computer's cores unsing function `mclapply` where the specified factor variables are re-leveled to each one of their levels, the `mer` model updated, and summaries returned. *MCMC p-value calculation is now implemented.* R will wait until all slave processes have finished running. See package `parallel` for more information about parallel computing. Note that tradional sequential computing can be achieved by specifying `mc.cores = 1`. Posthoc results can be viewed with function `summary.mcposthoc`.

Usage

```
mcposthoc.fnc(model, var, two.tailed = TRUE,
mcmc = FALSE, nsim = 10000, ndigits = 4, mc.cores = 1,
verbosity = 1, ...)
```

Arguments

`model`	A `mer` object (fitted by function `lmer`) or an `lm` object (fitted by function `lm`).
`var`	A named list of variable on which to perform the posthoc analysis. For example `list(ph1 = c("PronomOfTheme", "AnimacyOfRec", "DefinOfRec"), ph2 = c("SemanticClass"))`.
`two.tailed`	Logical. Whether to perform one- or two-tailed *t*-tests. Defaults to `TRUE`, i.e., two-tailed.
`mcmc`	Logical. Whether to calculate *p*-values using function `pamer.fnc` (the default) or using function `pvals.fnc` from package `languageR`.
`nsim`	An integer denoting the required number of Markov chain Monte Carlo samples. Defaults to 10000.
`ndigits`	Integer indicating the number of decimal places to be used in the *t* tables. Defaults to 4.
`mc.cores`	The number of cores to use, i.e. how many processes will be spawned (at most).
`verbosity`	Numeric. The amount of information printed to screen during the modeling process. The higher the number, the more information is printed. `0` turns this option off. Defaults to `1`.
`...`	Further arguments to pass to "mclapply".

Details

If `var = list(ph1 = c("PronomOfTheme", "AnimacyOfRec", "DefinOfRec"))`, for example, the function will re-level and update the model on each combination of the variable levels as follows:

```
(1) data$PronomOfTheme <- relevel(data$PronomOfTheme = "nonpronominal")
    data$AnimacyOfTheme <- relevel(data$AnimacyOfTheme = "animate")
    data$DefinOfTheme <- relevel(data$DefinOfTheme = "definite")

(2) data$PronomOfTheme <- relevel(data$PronomOfTheme = "nonpronominal")
    data$AnimacyOfTheme <- relevel(data$AnimacyOfTheme = "inanimate")
    data$DefinOfTheme <- relevel(data$DefinOfTheme = "definite")

(3) data$PronomOfTheme <- relevel(data$PronomOfTheme = "nonpronominal")
    data$AnimacyOfTheme <- relevel(data$AnimacyOfTheme = "animate")
    data$DefinOfTheme <- relevel(data$DefinOfTheme = "indefinite")

(4) data$PronomOfTheme <- relevel(data$PronomOfTheme = "pronominal")
    data$AnimacyOfTheme <- relevel(data$AnimacyOfTheme = "animate")
    data$DefinOfTheme <- relevel(data$DefinOfTheme = "definite")
```

```
(5) data$PronomOfTheme <- relevel(data$PronomOfTheme = "nonpronominal")
    data$AnimacyOfTheme <- relevel(data$AnimacyOfTheme = "inanimate")
    data$DefinOfTheme <- relevel(data$DefinOfTheme = "indefinite")

(6) data$PronomOfTheme <- relevel(data$PronomOfTheme = "pronominal")
    data$AnimacyOfTheme <- relevel(data$AnimacyOfTheme = "animate")
    data$DefinOfTheme <- relevel(data$DefinOfTheme = "indefinite")

(7) data$PronomOfTheme <- relevel(data$PronomOfTheme = "pronominal")
    data$AnimacyOfTheme <- relevel(data$AnimacyOfTheme = "inanimate")
    data$DefinOfTheme <- relevel(data$DefinOfTheme = "indefinite")

(8) data$PronomOfTheme <- relevel(data$PronomOfTheme = "pronominal")
    data$AnimacyOfTheme <- relevel(data$AnimacyOfTheme = "inanimate")
    data$DefinOfTheme <- relevel(data$DefinOfTheme = "definite")
```

On a cluster, instead of using `mcposthoc.fnc` it is better (faster and less complicated) to write a bash script that will create (1) `.R` files that will relevel the conditions and update the model, and (2) an associated `.sh` job submission script to submit the `.R` files. For example, let's consider two ERP analyses (regular past tense inflection and phrase structure) with three time windows each (300–400 ms, 550–700 ms, 750–850 in the regular past tense analysis, and 300–400 ms, 400–600 ms, and 750–850 ms in the phrase structure analysis). We investigated the effects of proficiency on ERP amplitudes. The initial models included a four-way interaction between Region of Interest (ROI) – with levels left anterior, left central, left posterior, midline anterior, midline central, midline posterior, right anterior, right central, and right posterior) – Group (with levels L1 and L2), Condition (wth levels control and violation), and Proficiency. After back-fitting the fixed effects, forward-fitting randomg effects, and reback-fitting the fixed effects as per `fitLMER.fnc`, the four-way interaction remained in every model. See Newman et al. (In preparation) for more details. The posthoc analysis script named `posthocs.sh` we used on the ACEnet cluster is as follows:

```
 time=(Reg300400 Reg550700 Reg750850 PS300400 PS400600 PS750850)
 condition=(Good Bad)
 group=(L1 L2)
 roi=(Lant Lcent Lpost Mant Mcent Mpost Rant Rcent Rpost)

for t in ${time[*]}; do for i in ${condition[*]}; do for j in ${group[*]}; do for k in ${roi[*]}; do
   ### create .R file where the modell is updated on the data where
   ### re-leveld on each possible combination of variable levels
   export CONDITION=$i;
   export GROUP=$j;
   export ROI=$k;
 echo 'condition<-Sys.getenv("CONDITION")' > "ph"$t$CONDITION$GROUP$ROI".R"
   echo 'group<-Sys.getenv("GROUP")' >> "ph"$t$CONDITION$GROUP$ROI".R"
   echo 'roi<-Sys.getenv("ROI")' >> "ph"$t$CONDITION$GROUP$ROI".R"
   echo 'load("models/m1'$t'.rda")' >> "ph"$t$CONDITION$GROUP$ROI".R"
   echo 'dat<-m1@frame' >> "ph"$t$CONDITION$GROUP$ROI".R"
 echo 'dat$Condition<-relevel(dat$Condition,'condition')' >> "ph"$t$CONDITION$GROUP$ROI".R"
 echo 'dat$Group<-relevel(dat$Group,'group')' >> "ph"$t$CONDITION$GROUP$ROI".R"
  echo 'dat$ROI<-relevel(dat$ROI,'roi')' >> "ph"$t$CONDITION$GROUP$ROI".R"
```

```
    echo 'm1<-update(m1,.~.,data=dat)' >> "ph"$t$CONDITION$GROUP$ROI".R"
echo 'save(m1,file="ph'$t$CONDITION$GROUP$ROI'.rda")' >> "ph"$t$CONDITION$GROUP$ROI".R"

    ### create the job submission script for the .R file created above
    echo '#$ -S /bin/bash' > "job.ph"$t$CONDITION$GROUP$ROI".sh"
    echo '#$ -cwd' >> "job.ph"$t$CONDITION$GROUP$ROI".sh"
    echo '#$ -j y' >> "job.ph"$t$CONDITION$GROUP$ROI".sh"
    echo '#$ -l h_rt=48:00:00' >> "job.ph"$t$CONDITION$GROUP$ROI".sh"
    echo '#$ -l h_vmem=8G' >> "job.ph"$t$CONDITION$GROUP$ROI".sh"
    echo '#$ -R y' >> "job.ph"$t$CONDITION$GROUP$ROI".sh"
echo '#$ -N "ph'$t$CONDITION$GROUP$ROI'"' >> "job.ph"$t$CONDITION$GROUP$ROI".sh"
echo 'R -q -f ph'$t$CONDITION$GROUP$ROI'.R' >> "job.ph"$t$CONDITION$GROUP$ROI".sh"

    ### submit the job
    qsub  "job.ph"$t$CONDITION$GROUP$ROI".sh"
done; done; done; done
```

and then type in the console

```
. posthocs.sh
```

On the ACEnet cluster, this results in 2 * 3 * 9 * 2 * 2 = 216 independent analyses, simultaneously using a total of 216 cores and 1728 GB of RAM. This posthoc analysis completes in about 3-6 hours.

Value

An object of class "mcposthoc" with the following slots:

`n`	The number of data points in data frame `data`.
`var`	A named list containing the names of the variables used in the posthoc.
`summaries`	A named list containing the posthoc summaries for each factor re-leveling. If `mcmc = FALSE`, data frames with upper- and lower-bound (anti-conservative and conservative, respectively) *df*s, *p*-values, and deviance explained (%) for each model term. If `mcmc = TRUE`, data frames with the estimated coefficients, their MCMC mean, the HPD 95 and the probability based on the *t* distribution with the number of observations minus the number of fixed-effects coefficients as degrees of freedom. This last *p*-value is anti-conservative, especially for small data sets.

warning

Parallel computing capabilities will not be available on Windows because `mclapply` relies on forking. Sequential computing, however, will work on Windows if `mc.cores = 1` (the default).

Note

It is not possible anymore to get p-values with function `pvals.fnc` of package `languageR`. Please see `http://stackoverflow.com/questions/19199713/lme4-and-languager-compatibility-error-input-model-i` for other possible avenues to get p-values.

Author(s)

Antoine Tremblay, Statistics Canada, <`trea26@gmail.com`>.

See Also

`summary.mcposthoc`

Examples

```
# see example in LMERConvenienceFunctions help page.
```

`pamer.fnc` *ANOVA with upper- and lower-bound* p*-values and R-sqaured values for LMER.*

Description

Compute upper- and lower-bound *p*-values for the analysis of variance (or deviance) as well as the amount of deviance explained (%) for each fixed-effect of an LMER model. Note that for `glmer` models, there is no deviance explained column.

Usage

```
pamer.fnc(model, ndigits = 4)
```

Arguments

`model`	A `mer` object (fitted by function `lmer`). Note that, at the moment, this function cannot be used with generalized linear mixed-effects models (`glmers`).
`ndigits`	Integer indicating the number of decimal places to be used in the ANOVA table.

Details

Upper-bound *p*-values are computed by using as denominator *df* `nrow(model@frame) - qr(model@X)4rank` (i.e., number of data points minus number of fixed effects including the intercept), which are anti-conservative. Lower-bound *p*-values are computed by using as denominator *df* `nrow(model@frame) - qr(model@X)4rank - number of random effects` (e.g., if by-subject intercepts and slopes, and there are 10 subjects, `10 * 2 = 20`). The amount of deviance explained by each model term (i.e., eta squared) is calculated as `[Sum of Squares for the effect] / [Sum of Squares total]`. More specifically: `as.data.frame(anova(model))[,2] / sum((model@frame[, dv]-mean(model@frame[, dv]))^2)` where `dv` is a vector of the names of the independent variables in the model.

Value

This function returns an object of class `data frame` with upper- and lower-bound (anti-conservative and conservative, respectively) *df*s, *p*-values, and deviance explained (%) for each model term. Note that for `glmer` models, there is no deviance explained column.

Author(s)

Antoine Tremblay, Statistics Canada, <`trea26@gmail.com`>

References

`[R] lmer, p-values` and `all that` available at https://stat.ethz.ch/pipermail/r-help/2006-May/094765.html.

Examples

```
# see example LMERConvenienceFunctions help page.
```

`perSubjectTrim.fnc` *Per-subject Trimming of Response Variable.*

Description

For each subject, removes data points that are, e.g., 2.5 standard deviations above or below the subject mean.

Usage

```
perSubjectTrim.fnc(data, response, subject, trim = 2.5)
```

Arguments

`data`	The data frame containing the data to be trimmed.
`response`	The quoted name of the column containing the to-be-trimmed data.
`subject`	The quoted name of the column contain subject identifiers.
`trim`	Threshold at which data points will be removed. Defaults to 2.5 (standard deviations above and below each subject's mean).

Value

The function returns the following objects:

`data`	The data with outliers removed.
`data0`	The original data prior to removing the outliers.
`n.removed`	The number of data points removed.
`percent.removed`	The percentage of removed data points.

Author(s)

Antoine Tremblay, Statistics Canada, <`trea26@gmail.com`>.

See Also

`mcp.fnc` `romr.fnc`

Examples

```
## Not run:
if("LCFdata" %in% .packages(all.available=TRUE)){
data(eegWide)
dat<-eegWide
rm(eegWide)
gc(TRUE,TRUE)
# per subject trimming
dat <- perSubjectTrim.fnc(dat, response = "Fz",
      subject = "Subject", trim = 2.5)$data
# ......
# n.removed = 5130
# percent.removed = 1.584507
}

## End(Not run)
```

`plotDensity3d.fnc` *Kernel density estimation for two continuous variables.*

Description

The densities of two continuous variables is first computed using the `density` function from package `stats`. The outer product of the two densities is computed, which can be plotted as a contour map or a perspective plot.

Usage

```
plotDensity3d.fnc(x, y, plot.type = "contour", color = "terrain",
xlab = NULL, ylab = NULL, zlab = NULL, main = NULL, cex = 1,
lit = TRUE, theta = 0, phi = 0, bw = "nrd0", adjust = 1, kernel = c("gaussian",
"epanechnikov", "rectangular", "triangular", "biweight", "cosine",
"optcosine"), weights = NULL, window = kernel, width, give.Rkern = FALSE,
n = 50, from, to, cut = 3, na.rm = FALSE, ...)
```

Arguments

`x, y`	Numeric vectors.
`plot.type`	The type of plot to make. Can be any of `"contour"` (default) or `"persp"`.

`color`	The colour scheme to use for plots. One of "`topo`", "`heat`", "`cm`", "`terrain`", "`gray`" or "`bw`". Schemes "`gray`" and "`bw`" also modify the colors used.
`xlab`, `ylab`, `zlab`	Titles for the axes. N.B. These must be character strings; expressions are not accepted. Numbers will be coerced to character strings.
`main`	The main title on top of the plot.
`cex`	The size of label and main text.
`lit`	Logical, specifying if lighting calculation should take place on geometry.
`theta`	Angle defining the viewing direction. `theta` gives the azimuthal direction. Used only if `plot.type = "persp"`.
`phi`	Angle defining the viewing direction. `phi` gives the colatitude. Used only if `plot.type = "persp"`.
`bw`, `adjust`, `kernel`, `weights`, `window`, `width`, `give.Rkern`, `n`, `from`, `to`, `cut`, `na.rm`	See help page to function `density`.
`...`	Further arguments passed to functions `image`, `contour` or `persp`.

Details

See help page to the `density` function.

Value

Either a contour map or a perspective plot. Invisibly returns

`x`	The numeric vector supplied in argument `x`.
`y`	The numeric vector supplied in argument `y`.
`xd`	The density object tied to vector `x`.
`yd`	The density object tied to vector `y`.
`mat`	The outer product of the `x` and `y` densities in matrix format.
`col`	The color used for plotting.

Author(s)

Antoine Tremblay, Statistics Canada, <`trea26@gmail.com`>.

See Also

`contour`; `persp`; `density`; `outer`

Examples

```
# see example in LMERConvenienceFunctions help page.
```

`plotLMER.fnc` *plot a mer object*

Description

Plot partial effects of a (generalized) linear mixed-effects model fit with `lmer` (compatible with package `lme4` version > 1.0).

Usage

```
plotLMER.fnc(model, xlabel = NA, xlabs = NA, ylabel = NA,
ylimit = NA, ilabel = NA, fun = NA, pred = NA, control = NA,
ranefs = NA, n = 100, intr = NA,lockYlim = TRUE, addlines = FALSE,
withList = FALSE, cexsize = 0.5, linecolor = 1,
addToExistingPlot = FALSE, verbose = TRUE, ...)
```

Arguments

`model`	a `mer` model object
`xlabel`	label for X-axis (if other than the variable name in the original model formula)
`xlabs`	character vector with labels for X-axes in multipanel plot (if other than the variable names in the original model formula); if used, `xlabel` should not be specified
`ylabel`	label for Y-axis (if other than the variable name of the dependent variable in the original model formula)
`ylimit`	range for vertical axis; if not specified, this range will be chosen such that all data points across all subplots, including HPD intervals, will be accommodated
`ilabel`	label for the interaction shown in the lower right-hand margin of the plot, overriding the original variable name in the model formula
`fun`	a function to be applied for transforming the dependent variable, if `NA`, no transformation is applied; for models with `family = "binomial"`, fun is set to `plogis` by default; this can be disabled by setting `fun=function(x)return(x)`.
`pred`	character string with name of predictor; if specified, a single plot will produced for the partial effect of this specific predictor
`control`	a two-element list `list(predictor, val)` specifying a `predictor` the value of which has to be set to `val` in the partial effect plot(s); the predictor name should be exactly as specified in `names(model@fixef)`. It is up to the user to make sure that name and value make sense, the code here hands full 'control' to the user.
`ranefs`	a four-element list `Group, Level, Predictor`, specifying a random-effect Group (e.g. `Subject`), a level (e.g., `S10`) and a value (e.g., `LogFrequency`) for which partial effects have to be calibrated.
`n`	integer denoting number of points for the plot, chosen at equally spaced intervals across the empirical range of the predictor variable

`intr`	a list specifying an interaction to be graphed; obligatory arguments are (1) the name of the interaction variable, followed by (2) a vector of values for that variable, followed by (3) the position for interaction labels (’"beg"’, ’"mid"’, or ’"end"’, or ’NA’ if no labels are desired), optionally followed by (4) a list with as first element a vector of colors and as second element a vector of line types. The number of elements in both vectors should match the number of values specified under (2) for the interaction predictor.
`lockYlim`	logical specifying whether all subplots should have the same range of values for the vertical axis; if `TRUE`, this range will be chosen to accomodate all fitted values including HDP intervals for all predictors across all plots
`addlines`	if TRUE, adds line(s) between levels of same factor(s)
`withList`	logical, if `TRUE`, a list will be output with all data frames for the subplots
`cexsize`	character expansion size (cex) for additional information in the plot for interactions
`linecolor`	color of lines in the plot, by default set to 1 (black)
`addToExistingPlot`	default FALSE, if set to TRUE, plot will be added to previous plot, but only if pred is specified
`verbose`	if TRUE (default), effect sizes and default transformations are reported
`...`	further graphical parameters to be passed down; warning: `col`, `pch`, `lty` and `cex` will often generate an error as they are internally already fully specified for specialized subplots

Details

When no predictor is specified, a series of plots is produced for the partial effects of each predictor. The graphs are shown for the reference level for factors and are adjusted for the median value for the other numerical predicors in the model. Interactions are not shown. The user should set up the appropriate number of subplots on the graphics device before running plotLMER.fnc().

Instead of showing all predictors jointly, plotLMER.fnc() can also be used to plot the partial effect of a specific predictor. When a specific predictor is specified (with `pred = ...`), a single plot is produced for that predictor. In this case, the `intr` argument can be used to specify a single second predictor that enters into an interaction with the selected main predictor.

Polynomials have to be fitted with `poly(..., degree, raw=TRUE)` and restricted cubic splines with `rcs()` from the `rms` package.

Note that any MCMC capabilities available in the `languageR` version of this function are not available in this version.

Value

A plot is produced on the graphical device.

Note

This code needs much more work, including (i) extension to `poly` with `raw=FALSE`, and (ii) general clean-up of the code.

Author(s)

R. H. Baayen, tweaked by Antoine Tremblay

See Also

`plotLMER3d.fnc`.

Examples

```
# see example in LMERConvenienceFunctions help page.
```

`plotLMER3d.fnc` *Dynamic 3d plot for* `mer` *object.*

Description

Plot dynamic 3d partial effects of a (generalized) linear mixed-effects model fit with `LMER`.

Usage

```
plotLMER3d.fnc(model = NULL, pred, intr, plot.type = "contour",
xlim = range(x, na.rm = TRUE), ylim = range(y, na.rm = TRUE),
zlim = range(z, na.rm = TRUE), xlab = NULL,
ylab = NULL, zlab = NULL, main = NULL, shift = 0, scale = 1, cex = 1,
fun = NA, n = 30, color = "topo", theta = 0, phi = 0,
contourstepsize = 0.2, legend.args = NULL, rug = FALSE,
plot.dat = "default", path = "default", ...)
```

Arguments

`model`	A `mer` object or `NULL` (the default) to plot from an existing data-plotting object returned by this function and saved as an `.rda` file.
`pred`	The quoted name of a model predictor.
`intr`	The quoted name of a continuous model predicor.
`plot.type`	The type of plot to make. Can be any of `"contour"` (default), or `"image.plot"` if package `fields` is available, `"persp"`
`xlim, ylim, zlim`	*x*-, *y*- and *z*-limits. The plot is produced so that the rectangular volume defined by these limits is visible.
`xlab, ylab, zlab`	Titles for the axes. N.B. These must be character strings; expressions are not accepted. Numbers will be coerced to character strings.
`main`	The main title on top of the plot.
`shift`	Constant to add to the smooth (on the scale of the linear predictor) before plotting. Defaults to 0. Passed to `plotRaw3d.fnc`.
`scale`	Constant by which to multiply the smooth before plotting. Defaults to 1. Passed to `plotRaw3d.fnc`.

`cex`	The size of label and main text.
`fun`	A function to be applied for transforming the dependent variable, if `NA`, no transformation is applied; for models with `family = "binomial"`, fun is set to `plogis` by default; this can be disabled by setting `fun=function(x)return(x)`.
`n`	Integer denoting number of points for the plot, chosen at equally spaced intervals across the empirical range of the predictor variable.
`color`	The colour scheme to use for plots. One of `topo`, `heat`, `cm`, `terrain`, `gray` or `bw`. Schemes `gray`, `grey`, and `bw` also modify the colors used.
`theta`	Angle defining the viewing direction. `theta` gives the azimuthal direction. Used only if `plot.type = "persp"`.
`phi`	Angle defining the viewing direction. `phi` gives the colatitude. Used only if `plot.type = "persp"`.
`contourstepsize`	The size of the steps from contour line to contour line.
`legend.args`	When `plot.type = "image.plot"`, arguments for a complete specification of the legend label. This is in the form of list and is just passed to the mtext function. (See example in `image.plot` help page). Defaults to `NULL`.
`rug`	Whether a rug ought to be plotted on the 3d surface. Defaults to `FALSE`.
`plot.dat`, `path`	Whether to cache the plotting data generated by a previous call to `plotLMER3d.fnc`. Generating the 3d plots can be time consuming. If the `plot.dat` argument is non-FALSE, the plotting information generated in the first call to the function will be saved so that in a second call to the function with exactly the same argument values, the plotting information will be retrieved and plotting will be significantly quicker. If `plotting.data = "default"` and `path = "default"`, the plotting information will be saved in a temporary directory and the name of the file containing the information will equal to `paste("lmer___",model@call,pred,intr,".rda",sep="` The name of the file and the path where it will be saved can be set by the user in the `plot.dat` and `path` arguments. For example, `plot.dat = "my_plotting.data"`, `path = "Documents"`. Note that `"lmer___"` will be appended to the begining of whatever is specified in `plot.dat` and `".rda"` to the end. Also note that if the user wants to save the plotting information returned by this function, the name of this object has to be `z`.
`...`	Further arguments to be passed to `image`, `contour`, `image.plot`, `persp`, or `persp3d`.

Details

See help page to Harald Baayen's `plotLMER.fnc` function as well as to Duncan Murdoch's `persp3d` function and the help page to function `image.plot` from package `fields`. To save screenshots of "persp3d" plots (after plotting), use function `rgl.snapshot` (produces png files) or function `rgl.postscript` (produces eps files).

Value

Invisibly returns plotting information (`x` and `y` vectors, `z` matrix, and colors, `col`). If `plot.type = "contour"`, `plot.type = "image.plot"`, or `plot.type = "persp"`, a contour or perspective plot, respectively. If `ret = TRUE`, a two-element list is returned containing the *matrix* and the matrix of corresponding colors is returned. If argument `intel` in non-null, a file containing plotting information will be saved.

Author(s)

Antoine Tremblay, Statistics Canada, <`trea26@gmail.com`>.

See Also

`persp`; `contour`; `plotLMER.fnc`.

Examples

```
if(try(require(LCFdata,quietly=TRUE))){
 data(z)
 temp.dir <- tempdir()
 save(z,file=file.path(temp.dir,"lmer___z.rda"))

  plotLMER3d.fnc(pred = "LengthBc", intr = "WMCc",
    plot.dat = "z", path = temp.dir)
  plotLMER3d.fnc(pred = "LengthBc", intr = "WMCc",
    plot.type = "persp", phi = 25, plot.dat = "z",
    path = temp.dir)
}
```

`plotRaw3d.fnc` *Visualize raw surface averages (3d)*

Description

For a specified response variable and interacting continuous predictors, visualize in 3d the surface average.

Usage

```
plotRaw3d.fnc(data = NULL, response = NULL, pred = NULL, intr = NULL,
xy = TRUE, color = "topo", zlim = NULL, xlab = NULL, ylab = NULL,
zlab = NULL, main = NULL, shift = 0, scale = 1, plot.type = "contour",
theta = 30, phi = 30, ticktype = "detailed",
contourstepsize = 1, legend.args = NULL, ...)
```

Arguments

`data`	A data frame.
`response`	The quoted name of a continuous response variable.
`pred`	The quoted name of a continuous predictor.
`intr`	The quoted name of an interacting continuous predictor.

`xy`	Whether to the `x` and `y` values from the data or to set them to `seq(0, 1, len = nrow(z))`. Defaults to `TRUE`.
`color`	The colour scheme to use. One of `"topo"`, `"heat"`, `"cm"`, `"terrain"`, `"gray"` or `"bw"`.
`zlim`	A two element vector specifying the plotting limits for the z-axis.
`xlab`, `ylab`, `zlab`	Titles for the axes. N.B. These must be character strings; expressions are not accepted. Numbers will be coerced to character strings.
`main`	The main title on top of the plot.
`shift`	Constant to add to the smooth (on the scale of the linear predictor) before plotting. Defaults to 0.
`scale`	Constant by which to multiply the smooth before plotting. Defaults to 1.
`plot.type`	The type of plot to make. Can be any of `"contour"` or `"persp"`.
`theta`	Angle defining the viewing direction. `theta` gives the azimuthal direction.
`phi`	Angle defining the viewing direction. `phi` gives the colatitude.
`ticktype`	Character: `"simple"` draws just an arrow parallel to the axis to indicate direction of increase; `"detailed"` draws normal ticks as per 2D plots.
`contourstepsize`	The size of the steps from contour line to contour line. Defaults to 1. Used only if `plot.type = "contour"`.
`legend.args`	When `plot.type = "image.plot"`, arguments for a complete specification of the legend label. This is in the form of list and is just passed to the mtext function. (See example in `image.plot` help page). Defaults to `NULL`.
`...`	Further arguments passed to functions `image`, `image.plot`, `contour` or `persp`.

Details

`NA`s will be set to `0`.

Value

Either a dynamic 3d perspective plot, a perspective plot, or a contour plot. Also invisibly returns the plotting matrix and the color vector.

Author(s)

Antoine Tremblay, Statistics Canada <`trea26@gmail.com`>

Examples

```
# see example in LMERConvenienceFunctions help page.
```

---

relLik *Relative log-likehood.*

Description

Calculate the relative log-likehood between two models.

Usage

```
relLik(x, y, method = c("AIC", "BIC"), ndigits = 6, ...)
```

Arguments

`x, y`	Fitted model objects for which there exists a `logLik` method to extract the corresponding log-likelihood, or objects inheriting from class `logLik`.
`method`	Whether to base the comparison on `AIC` or `BIC`. Defaults to "AIC".
`ndigits`	An integer denoting the number of decimal digits in the output.
`...`	Further arguments to pass to `AIC` or `BIC`.

Details

The relative log-likelihood is calculated as `exp((abs(AIC(x) - AIC(y)))/2)` or `exp((abs(BIC(x) - BIC(y)))/2)`, depending on the method.

You can find information regarding differences between AIC and BIC from `http://methodology.psu.edu/eresources/as`

Value

A vector with values:

`AIC(x), BIC(x)`	The `AIC` or]codeBIC value of the first model object.
`AIC(y), BIC(y)`	The `AIC` or]codeBIC value of the second model object.
`relLik`	The relative likelihood between the two models. Model `y` will be that much more likely given the data than model `x`.

Author(s)

Antoine Tremblay, Statistics Canada, <`trea26@gmail.com`>

References

On AIC and relative log-likelihood (which they call evidence ratio):

Symonds, M.R.E and Moussalli, A. (2011). *A brief guide to model selection, multimodel inference and model averaging in behavioural ecology using Akaike's information criterion. Behavioral Ecology and Sociobiology*, 65, 13–21. doi: 10.1007/s00265-010-1037-6

See Also

`logLik`; `AIC`; `BIC`

Examples

```
# see example in LMERConvenienceFunctions help page.
```

`romr.fnc` *Exclude outliers.*

Description

Exclude outliers with a standardized residual at a distance greater than 2.5 standard deviations from 0. Note that this function cannot be used with generalized linear mixed-effects models (`glmer`s).

Usage

```
romr.fnc(model, data, trim = 2.5)
```

Arguments

`model`	A `mer` object (fitted by function `lmer`). Note that this function cannot be used with generalized linear mixed-effects models (`glmer`s).
`data`	The data frame on which the `mer` object was fitted.
`trim`	Threshold at which residuals will be removed. Defaults to 2.5 (standard deviations above and below the residuals mean).

Value

The function returns the following objects:

`data`	The data with outliers removed.
`data0`	The original data prior to removing the outliers.
`n.removed`	The number of data points removed.
`percent.removed`	The percentage of removed data points.

Author(s)

Antoine Tremblay, Statistics Canada, `<trea26@gmail.com>`, with contrbutions from Andy Flies, Michigan State University.

References

Baayen, R.H. (2008). *Analyzing Linguistic Data. A Practical Introduction to Statistics Using R*. Cambridge, UK: Cambridge University Press.

Newman, A.J., Tremblay, A., Nichols, E.S., Neville, H.J., and Ullman, M.T. (submitted). The Influence of Language Proficiency on Lexical-Semantic Processing in Native and Late Learners of English: ERP evidence. Submitted to the *Journal of Cognitive Neuroscience*.

Tremblay, A. and Tucker B. V. (submitted). What can the production of four-word sequences tell us about the mental lexicon? Submitted to *The Mental Lexicon*.

See Also

`mcp.fnc` `perSubjectTrim.fnc`

Examples

```
# see example in LMERConvenienceFunctions help page.
```

`summary.mcposthoc` *Summarize a "mcposthoc" object.*

Description

This function extracts the desired portions of an "mcposthoc" object.

Usage

```
## S3 method for class 'mcposthoc'
summary(object, ph.list = NULL,
term = NULL, print = TRUE, ...)
```

Arguments

`object`	An "mcposthoc" object as returned by function `mcposthoc.fnc`.
`ph.list`	The name of the posthoc analysis for which results are desired. For example, if, in function `mcposthoc.fnc`, argument `var` was set to `list(ph1 = c("PronomOfTheme", "AnimacyOfRec", "DefinOfRec"), ph2 = c("SemanticClass"))`, `ph.list` should be one of "ph1" or "ph2". Defaults to `NULL`. If `ph.list = NULL` and more than one posthoc analysis was performed, the user will be prompted to select one of the analyses.
`term`	The model term for which posthoc results are desired. Defaults to `NULL`, in which case the user will be prompted to select a term.
`print`	Whether to print to screen the posthoc summary. Defaults to `TRUE`.
`...`	Not used.

Details

The function creates a summary data frame from statistics obtained from an "mcposthoc" object for the specified term. It goes through each element of the `ph.list` – each list element is the summary of the model re-leveled on one factor level (or combination of factor levels) – extracts the row corresponding to the `term`, and binds it to the other extracted rows.

Value

`ph.list`	The posthoc list in the "mcposthoc" object from which the summary originates.
`term`	The term from the posthoc list for which a summary is desired.
`summary`	The posthoc summary.

Author(s)

Antoine Tremblay, Statistics Canada, <`trea26@gmail.com`>

See Also

`mcposthoc.fnc`; `pamer.fnc`.

Examples

```
### See examples from mcposthoc.fnc() help page.
```

Index