EDF9850_Functions.R

A User's Guide to the Custom R Functions for This Course

Written for readers who are new to R and to multilevel/mixed-effects models
Compiled by Claude Code under the direction of Lesa Hoffman
last updated 11/06/26

Table of Contents

Getting Started 3
A Few Terms Used Throughout This Guide 3
univSummary() 4
corrSummary() 5
freqSummary() 6
addUnitMeans() 7
lmerSummary() 7
glsSummary() 12
lmeSummary() 13
glmerSummary() 18
clmmSummary() 21
clmSummary() 24
con1DSummary() 27
con1DTable() 29
emmSat() 31
conMDSummary() 32
PseudoR2() 35
TotalR2() 36
randLRT() 36
glmerRanova() 37
plotResiduals() 39
marginalV() 40

Getting Started

This guide explains what each function in EDF9850_Functions.R does, what its options mean, and when to use it. Each entry explains not just how to call the function, but what the statistics it prints actually mean, since these functions assume no prior R experience.

Loading the functions

Before using any function below, run this once at the start of your R session (adjust the path to wherever you saved the file, or load it directly from the web):

```
source("EDF9850_Functions.R")
source("https://www.lesahoffman.com/EDF9850/EDF9850_Functions.R") # From the web
```

This also loads the R packages these functions depend on—nlme, lme4, lmerTest, performance, emmeans, and ordinal—installing them automatically the first time if they are not already on your computer, without the usual "masked" package-loading notices some of them would otherwise print (they are harmless naming overlaps between these packages, not something to act on).

Helpful error messages

If you call one of these functions on an object that does not exist yet (for example, a typo in a model's name, or a model you have not fit yet), or on the wrong kind of model (for example, handing an lmer model to glsSummary()), you will get a plain-English error message explaining the problem, instead of a confusing low-level R error. Just fix the problem the message describes and run the line again—nothing else is required.

Significance stars (optional)

By default these functions print p-values only. To add significance stars to every p-value column in every function at once, run this once in your R session:

```
options(EDF9850.signif.stars = TRUE)     # turn stars on for all of these
functions
options(EDF9850.signif.stars = FALSE)    # turn them back off (the default)
```

Stars use R's usual cutoffs: *** for p < .001, ** for p < .01, * for p < .05, and a period (.) for p < .1. This setting is separate from R's own show.signif.stars option, so it is unaffected by how that is set. In each table with p-values, the stars go in their own unlabeled column at the end of each row, after all the other columns (in corrSummary()'s correlation matrix, which has no single row to end, each star follows its own p-value instead). The likelihood-ratio (LRT) tables never get stars, in any function, and the saved/returned objects are never changed—stars affect only what is printed.

A Few Terms Used Throughout This Guide

These functions report several quantities that are specific to multilevel/mixed-effects models. Brief plain-language definitions are collected here so each function's description below can stay short.

- **Fixed effects** Model terms (the intercept and predictors) whose single estimated value is assumed to apply to every person/observation in the data.
- **Random effects** The extent to which the intercept (and, if modeled, one or more slopes) varies from cluster to cluster (e.g., from school to school).

- **ICC1** Intraclass correlation: the proportion of the total outcome variance that lies between clusters rather than within them.
- **ICC2** The reliability of the cluster means—how trustworthy a given cluster's average is as an estimate of that cluster's true mean.
- **LRT** Likelihood ratio test: a significance test for whether a more complex model fits significantly better than a simpler, nested model.
- **Mixture p-value (p.mix)** A corrected p-value for an LRT that removes a variance component. Because a variance cannot go below 0, its true null distribution is a 50:50 mixture of chi-square(df) and chi-square(df−1), not plain chi-square(df) (Self & Liang, 1987)—for a 1-df variance test this works out to exactly half the ordinary p-value. It appears alongside the regular p-value in every LRT table these functions produce.
- **Design effect / Effective N** The design effect is how much a cluster sample's precision is degraded by clustering, computed as 1 + (mean cluster size − 1) × ICC1. The effective N is the total sample size divided by the design effect—the "equivalent" sample size if the data were not clustered at all.
- Slope reliability (SR) The reliability of a random slope: how trustworthy a given cluster's own slope is as an estimate of that cluster's true slope. Computed as slope variance / (slope variance + residual variance / (mean L1n × variance of the level-1 predictor)), where L1n is the number of level-1 observations per cluster; the predictor should be centered within cluster. Caveat: the residual-variance term approximates each cluster's own slope sampling variance as if the predictor's within-cluster sum of squares were a fixed, known quantity; in practice it varies from cluster to cluster, which understates that sampling variance somewhat (more so in small clusters) and inflates SR a little as a result. It also assumes the predictor is uncorrelated with any other predictor sharing the same random-effects block, and does not apply at all, even approximately, to a random slope on an interaction/product term (e.g. a random X1:X2 slope)—treat SR for those with real caution.
- **Satterthwaite df** An approximate denominator degrees-of-freedom used for each fixed effect's significance test. Unlike ordinary regression, multilevel models do not have one simple, exact df, so this approximation is used instead.
- **Cohen's d / partial r** Two standardized effect-size measures for a fixed effect, letting you judge its size on a common scale regardless of the outcome's original units.
- **Odds ratio / rate ratio** The GLMM (glmerSummary()) equivalent of Cohen's d for a Wald z-test: exp(Estimate). For a logit-link (e.g. logistic) model this is an odds ratio; for a log-link (e.g. Poisson) model, a rate ratio. A value of 1 means no effect; above 1 means the outcome becomes more likely/frequent as the predictor increases, below 1 means less.
- **CI** Confidence interval: a range of plausible values for a parameter, at a chosen confidence level (95% by default in these functions).
- **Pseudo-R2** The percent reduction in one variance component (e.g., the random intercept variance) when predictors are added, comparable in spirit to R2 in ordinary regression but computed separately for each variance component.
- **Total R2** The percent of the total outcome variance explained by a model's fixed effects only.

univSummary()

Basic descriptive statistics (N, Mean, SD, Variance, Min, Max, Median) for one or more variables, optionally split by group

Reports the seven most basic descriptive statistics for one or more numeric variables in a data frame: N (the non-missing sample size), Mean, SD (standard deviation), Variance, Min, Max, and Median. This is a simpler, more focused alternative to functions like psych::describe() when this basic summary is all you need. Optionally split the results by one or more grouping variables (similar to psych::describeBy()), which produces one row per variable-by-group combination in a single table rather than a separate table for each group. The Median is the middle observation when N is odd; when N is even, medianType (see below) controls how the two middle observations are combined.

Basic usage

```
univSummary(MyData)
univSummary(MyData, vars=c("mpg","hp"), by="cyl")
```

Arguments

Argument	Default	What it does
`data`	(required)	The data frame containing the variable(s) to summarize.
`vars`	NULL	The name(s) (in quotes, combined with c() for more than one) of the numeric column(s) to summarize. If left NULL, every numeric column in data is summarized.
`by`	NULL	The name(s) (in quotes, combined with c() for more than one) of one or more grouping column(s) in data. If left NULL, statistics are computed overall with no grouping; if provided, statistics are computed separately for every combination of these groups.
`na.rm`	TRUE	Whether to drop missing values before computing statistics, done separately for each variable (and for each group, when by is used).
`digits`	3	The number of decimal places shown in the printed table.
`medianType`	7	How the Median is computed when N is even (with an odd N it is always the middle observation). A number from 1 to 9, the type= of R's quantile() function; the default, 7, is R's own default. Types 2 and 5 through 9 all give the average of the two middle observations, which is the usual median (the same as median()); types 1, 3, and 4 give the lower of the two.

corrSummary()

Bivariate (Pearson) correlations among two or more variables, with each p-value shown alongside its correlation

Reports a Pearson correlation matrix for two or more numeric variables in a data frame, together with the p-value testing whether each correlation differs from zero. By default (p.separate=FALSE), the table is stacked: each variable's row of correlations is followed immediately by an indented row giving that row's p-values in parentheses, so every p-value sits directly beneath its correlation. Set p.separate=TRUE to instead print the correlations and p-values as two separate plain tables, with no

stacking. Each pair of variables is correlated using only the rows where both variables are observed, so the sample size can differ from pair to pair when there is missing data.

Basic usage

```
corrSummary(MyData, vars=c("mpg","hp","wt"))
corrSummary(MyData, vars=c("mpg","hp","wt"), p.separate=TRUE)
```

Arguments

Argument	Default	What it does
`data`	(required)	The data frame containing the variable(s) to correlate.
`vars`	NULL	The name(s) (in quotes, combined with c() for more than one) of the numeric column(s) to correlate. If left NULL, every numeric column in data is used. At least two variables are required.
`p.separate`	FALSE	Set to TRUE to print the correlations and their p-values as two separate tables instead of one stacked table.
`digits`	3	The number of decimal places shown in the printed table.

freqSummary()

Frequencies (N) and proportions for one or more categorical variables, optionally split by group

Reports the frequency (N) and proportion of each observed level for one or more categorical variables in a data frame—analogous to univSummary(), but for categorical rather than numeric variables. Produces one row per variable-by-level combination in a single table (one row per variable-by-level-by-group combination when by is used), rather than a separate table for each variable. table()'s own level order is kept—a factor's own level order (including any level with zero occurrences, shown as N=0), or sorted order for a character/logical/numeric variable—and Proportion always sums to 1 within each variable (and group, when by is used).

Basic usage

```
freqSummary(MyData)
freqSummary(MyData, vars="lunch", by="schoolID")
```

Arguments

Argument	Default	What it does
`data`	(required)	The data frame containing the variable(s) to tabulate.
`vars`	NULL	The name(s) (in quotes, combined with c() for more than one) of the column(s) to tabulate. If left NULL, every factor, character, or logical column in data is used (the natural "categorical" columns); a numeric column can still be tabulated by naming it explicitly, e.g. for a 0/1/2-coded group variable.
`by`	NULL	The name(s) (in quotes, combined with c() for more than one) of one or more grouping column(s) in data. If left NULL, frequencies

Argument	Default	What it does
		are computed overall with no grouping; if provided, computed separately for every combination of these groups.
`useNA`	TRUE	Whether missing values get their own "NA" level (and their own row), done separately for each variable (and for each group, when by is used). Set to FALSE to drop them instead.
`digits`	3	The number of decimal places shown for Proportion in the printed table.

addUnitMeans()

Prepares raw data for a multilevel model by adding cluster-level information

Use this before fitting a model when you need cluster-level information added to your raw (level-1) data—most commonly, the number of level-1 units in each cluster and the cluster mean of one or more level-1 variables (a common way to create a level-2 "contextual" predictor, such as a school's mean SES). It returns a new data frame with these columns added to every row.

Basic usage

```
Example3 = addUnitMeans(data=Example3, unitVariable="schoolID",
                        meanVariables=c("ses"), newNames=c("CMses"))
```

Arguments

Argument	Default	What it does
`data`	(required)	The data frame containing your raw, level-1 data.
`unitVariable`	(required)	The name (in quotes) of the column that identifies the cluster each row belongs to, e.g. "schoolID".
`meanVariables`	(required)	The name(s) (in quotes, combined with c() for more than one) of the column(s) to compute a cluster mean for.
`newNames`	(required)	The name(s) to give the new cluster-mean column(s), in the same order as meanVariables.

What it adds: a column named "Nper<unitVariable>" giving the number of level-1 rows in that row's cluster, plus one new column per entry in meanVariables/newNames giving that cluster's mean on that variable, merged onto every row belonging to that cluster.

lmerSummary()

For multilevel models fit with lme4::lmer()

Prints a complete, course-formatted summary of a model fit with lmer(): the model fit statistics (AIC, BIC, log-likelihood, -2*logLik), the random effects table, an omnibus F-test that all fixed slopes are jointly zero, and the fixed effects table. Several additional pieces of output—intraclass correlations, a design effect and effective sample size, random slope reliability, random-effect confidence intervals,

standardized effect sizes, and a likelihood-ratio test for the random effects—can be turned on as needed. It also automatically creates new objects in your R environment holding each variance/covariance component from the model, so you do not need to look them up by hand for further calculations. If a random-effect variance is estimated at exactly 0 (a singular fit), every correlation involving it is undefined and prints as NaN, with a note under the random effects table; glmerSummary() and clmmSummary() do the same.

Basic usage

```
lmerSummary(Model1, name="Model 1: Empty Means Model", ICC1=TRUE, LRT=TRUE)
```

Arguments

Argument	Default	What it does
`model`	(required)	The fitted lmer model object, entered without quotes (e.g. Model1).
`name`	NULL	An optional text label (in quotes) printed above the output, e.g. to identify which model this is.
`ddf`	"Satterthwaite"	The method used for the fixed-effects tests' degrees of freedom: "Satterthwaite" (the recommended default), "Kenward-Roger", or "asymptotic" (treats df as infinite, i.e. a Wald z-test). The omnibus F-test uses the same method when it is "Kenward-Roger", Satterthwaite otherwise (no asymptotic option there). Effect sizes (below) are not available when ddf="asymptotic", since their formulas require a finite df.
`CI`	FALSE	Set to TRUE to add a confidence interval to the Fixed Effects table.
`levelCI`	95	The confidence level (%) used when CI=TRUE, e.g. levelCI=90 for a 90% CI.
`predMeans`	FALSE	Set to TRUE to print the model-predicted mean of every level of each factor predictor, or give the name(s) of just the factor(s) you want in quotes, e.g. predMeans="time_F". One table per factor (Level, Mean, SE, df, plus Lower/Upper when CI=TRUE) prints right after the Fixed Effects table. It works whatever the factor's coding is (indicator, sequential, and so on)— see the note below.
`residPlot`	FALSE	Set to TRUE to draw three plots of the level-1 residuals, side by side, for checking the normality and constant-variance assumptions: a histogram with a normal curve laid over it, a normal Q-Q plot (points close to the line = approximately normal), and the residuals against the fitted values with a zero line and a smoothed trend line (an even band around zero = constant variance; a funnel or curve = a problem). The residuals are the model's own conditional level-1 residuals, so the fitted values include the random effects. Instead of TRUE, give the name of a variable in the model's data in quotes, e.g. residPlot="time_F", to add a fourth plot of boxplots of the residuals at each level of that variable (similar box heights = constant variance across the levels)

Argument	Default	What it does
		and to print the residuals' N, mean, and SD at each level; the variable does not have to be in the model. Drawn after all the printed output; nothing is added to the printed tables.
`effectSizes`	FALSE	Set to TRUE to print Cohen's d and partial r for each fixed slope (requires df other than "asymptotic").
`ICC1`	FALSE	Set to TRUE to print the ICC1 intraclass correlation, both adjusted (the random intercept variance, or the sum of them if there are several grouping factors, over that plus the residual variance) and unadjusted (which also adds the variance of the fixed effects' predictions to the denominator, so it is smaller whenever the model has predictors). Both are computed directly from the variance components, so they do not depend on the performance package; only a model with random slopes uses performance::icc(). For a model with more than one random-intercept grouping factor (nested or crossed, e.g. classrooms within schools), it also prints a table headed "Proportion of Variance by Source (adj = adjusted after fixed effects):" with an adjusted proportion (each random intercept variance and the residual variance as a share of their total; random slope variances are not part of it) and an unadjusted proportion (which also has a "Fixed Effects" row, the variance of the fixed effects' predictions). If the grouping factors are nested, the table is ordered from the highest level down with a Level column (residual = level 1), and for a three-level nested model a second ICC is printed, L3 / (L2 + L3), the share of the upper-level variance that lies at the top (level 3) level, in an adjusted and an unadjusted version (the unadjusted one adds the fixed effects' variance to the denominator). See Neff, ICC2, and slopeRel below for how they handle such models. In such a model randCI also shows one row per grouping factor's random intercept, labeled with that factor.
`Neff`	FALSE	Set to TRUE to print the design effect (using the mean cluster size and ICC1) and the resulting effective sample size. For a model with more than one random-intercept grouping factor, the design effect adds up every level's contribution, 1 + the sum over levels of (mean n per unit at that level − 1) × that level's share of the total variance, and the line lists the mean n used at each level; this requires fully nested factors (it prints "not available" for crossed ones).
`ICC2`	FALSE	Set to TRUE to print the ICC2 reliability of the cluster means. For a model with fully nested random-intercept grouping factors, it prints one ICC2 per level (highest level first), each equal to that level's own variance over itself plus the noise from the lower levels averaged over the unit's size, e.g. for classrooms in schools: class = L2 / (L2 + residual / n per class), school = L3 / (L3 + L2 / classes per school + residual / n per school). It prints "not available" for crossed factors.

Argument	Default	What it does
`slopeRel`	FALSE	Set to TRUE to print the slope reliability (SR) of each random slope, along with the slope variance, residual variance, predictor variance, and mean L1n it is computed from (see the glossary above). Requires at least one random slope in the model. Each slope uses the mean cluster size (L1n) of its own grouping factor, so in a multi-level model a slope that varies across schools and one that varies across classrooms each get their own.
`randCI`	FALSE	Set to TRUE to print confidence intervals for the random effects, automatically paired with the fixed effect of the same name. Automatically set to TRUE if randCIfixed and randCIrandom are both specified. Each row is labeled with the random effect and its grouping factor, "Term | Group" (for example, "Intercept | schoolID"), the same style as the LRT table.
`randCIfixed / randCIrandom`	NULL	Use together to add one or more custom-paired CIs for a fixed effect and a random effect that do not share the same name—e.g. a level-2 predictor's CI on the random intercept. Enter the name of each predictor in quotes; pass equal-length vectors (e.g. randCIfixed=c("PaidvRed","UnpaidvRed"), randCIrandom=c("WC_PaidvRed","WC_UnpaidvRed")) to request more than one custom pairing at once, matched up by position. The intercept can be named Intercept, as the tables print it, or "(Intercept)".
`LRT`	FALSE	Set to TRUE to print a likelihood ratio test for the random effect(s) (via ranova()); the table's first column, "Random Effect Dropped", names what each row removes in a condensed "Term | Group" form (e.g. ranova()'s own "WC_PvD in (1 + WC_PvD + WC_RvF | schoolID)" is shown as "WC_PvD | schoolID"). The table includes both the ordinary p-value and the p.mix mixture-corrected p-value. A random intercept that shares its grouping factor with random slopes is not tested on its own (ranova() tests only the slopes of such a term); LRTdrop="Intercept" with LRTdropGroup adds that test. Automatically set to TRUE if LRTdrop is specified.
`LRTdrop`	NULL	Add one extra row to the LRT table testing a custom, user-chosen combination of random effects removed together, all from ONE random-effects dimension -- e.g. LRTdrop=c("WC_PvD","WC_RvF") to test two correlated random slopes jointly, or LRTdrop="Intercept" for just the random intercept ("(Intercept)", the name lme4 itself uses, also works). Useful because ranova()'s own rows only ever test one slope (or one correlation) at a time, so for a correlated multi-slope block they can't answer "do these slopes matter jointly?" on their own. Enter the name(s) in quotes (combined with c() for more than one), matching how each random effect is named in that model (as it appears in the Model for the Variance table); the row is labeled "Term(s) | Group", the same style as the table's other rows. Its p.mix is left blank: that

Argument	Default	What it does
		correction assumes dropping exactly one variance at a time, which doesn't generally hold for an arbitrary combination.
`LRTdropGroup`	NULL	Only needed if the same random-effect name in LRTdrop occurs in more than one random-effects dimension (e.g. WC_PvD as a slope under both schoolID and classroomID) -- name which one you mean (in quotes), matching that dimension's own grouping factor, e.g. LRTdropGroup="schoolID". Leave NULL when there's no such ambiguity; specifying it without a real ambiguity is harmless (LRTdrop is then only applied within that one dimension). An unmatched name in either argument stops with a clear error rather than guessing.
`digits`	3	The number of decimal places shown in the printed tables.
`saveObjects`	FALSE	Set to TRUE to save the fixed-effects table and every variance/covariance component from the model as their own named objects (see "What it saves automatically" below). Off by default.
`envir`	(automatic)	Where the objects below are stored when saveObjects=TRUE. Leave at its default.

Note: For any fixed effect using standard factor (treatment) coding, the Fixed Effects table also adds a row naming that predictor's reference category (e.g. group (ref = control)), placed immediately before its own rows.

Note: With predMeans, each level's predicted mean is the Intercept plus that level's own coding, so it comes out right for indicator coding, 0/1 sequential coding, and centered sequential (successive-difference) coding alike, without adding coefficients by hand. Every OTHER predictor is held at its coded zero point— numeric predictors at 0, other factors at their reference level (or at the grand mean under centered coding)— which is the same point the Intercept describes, so the means line up with the Fixed Effects table; the table heading says "(other predictors at 0)" when the model has any. The df follow the ddf argument (SE and df use the same method as the Fixed Effects table; ddf="asymptotic" gives df = Inf). In a no-intercept (cell-means) model a factor's predicted means equal its coefficients. The means are returned as predMeans in the saved result (one data frame per factor) but are not created as separate objects by saveObjects. A name that is not a factor predictor in the model stops with an error listing the model's factors.

What it saves automatically

When saveObjects=TRUE, using the model's own name (e.g. a model called Model1): Model1_VarCorr (the full variance-component table) and Model1_Fixed (the full fixed-effects table), plus one object per variance/covariance component, e.g. Model1_tau0 (random intercept variance), Model1_sigma2 (residual variance), and Model1_tau_<predictor> for any random slope variance. If the model has more than one grouping factor (e.g. students in classrooms in schools), each of those names also gets its grouping factor appended so the factors do not overwrite each other, e.g. Model1_tau0_school and Model1_tau0_class (a factor such as class:school appears as class.school).

glsSummary()

For single-level models fit with nlme::gls()

Same idea as lmerSummary(), but for a gls() model—a single-level regression model with no random effects at all. This is most often the comparison/baseline model without any clustering. Prints the model fit statistics, the residual variance, an omnibus F-test that all fixed slopes are jointly zero, and the fixed effects table. For a model fit with weights=varIdent(form=~1|group) (separate residual variances by group, e.g. by treatment arm or zygosity), it also automatically prints a Residual Variance by Group table—the single residual variance above is then only the reference group's. In a no-intercept model with a factor entered as cell means (e.g. y ~ 0 + group + x), that factor's coefficients are the group means, not slopes, so they are left out of the omnibus test that all slopes are zero. By default each fixed effect is given its own Satterthwaite df and the omnibus test a Satterthwaite denominator df (as lmeSummary() does), which matters for a gls model with an estimated residual covariance matrix (e.g., an unstructured R matrix from corSymm() with varIdent(), or an AR1 structure); use ddf="residual" for the older approach of N minus the number of fixed effects for every effect. The Fixed Effects table is followed by a line stating which denominator df method was used. For a model with a residual correlation structure (for example corCompSymm, corAR1, or corSymm), it also prints a Residual Correlation table—the structure, each parameter's name, and its estimate—right after the residual variance; for compound symmetry, Rho is the correlation among all of a person's residuals (the ICC of the equivalent random-intercept model).

Basic usage

```
glsSummary(Model0, name="Model 0: No Clustering", CI=TRUE)
```

Arguments

Argument	Default	What it does
`model`	(required)	The fitted gls model object, entered without quotes (e.g. Model0).
`name`	NULL	An optional text label (in quotes) printed above the output.
`CI`	FALSE	Set to TRUE to add a confidence interval to the Fixed Effects table.
`levelCI`	95	The confidence level (%) used when CI=TRUE.
`predMeans`	FALSE	Set to TRUE to print the model-predicted mean of every level of each factor predictor, or give the name(s) of just the factor(s) you want in quotes, exactly as the term is written in the model, e.g. predMeans="factor(occasion)". One table per factor (Level, Mean, SE, df, plus Lower/Upper when CI=TRUE) prints right after the Fixed Effects table, with the same df method as that table: each mean's own Satterthwaite df by default, or the residual df N minus the number of fixed effects with ddf="residual".
`effectSizes`	FALSE	Set to TRUE to print Cohen's d and partial r for each fixed slope.
`digits`	3	The number of decimal places shown in the printed tables.

Argument	Default	What it does
`saveObjects`	FALSE	Set to TRUE to save the fixed-effects table and the residual variance as their own named objects (see "What it saves automatically" below). Off by default.
`envir`	(automatic)	Where the objects below are stored when saveObjects=TRUE. Leave at its default.
`ddf`	"Satterthwaite"	The denominator degrees of freedom for the fixed effects. "Satterthwaite" (the default) gives each fixed effect its own approximate df (via the same emmeans-based method that lmeSummary() uses, computed with a fixed random seed so it is reproducible) and a Satterthwaite denominator df for the omnibus F-test; the p-values, confidence intervals, and effect sizes use those df, and the output states the method used under the Fixed Effects table. "residual" instead uses N minus the number of fixed effects for every term and for the omnibus F-test. If the Satterthwaite df cannot be computed, a warning is given and the residual df are used.

Note: glsSummary() has no ICC1, Neff, ICC2, randCI, or LRT arguments—a gls model has no random effects, so those concepts do not apply to it.

Note: With predMeans, each level's predicted mean is the Intercept plus that level's own coding, so it is right whatever the factor's coding is, exactly as described for lmerSummary() above, with every other predictor held at its coded zero point. For a model such as y ~ 1 + factor(occasion) with an unstructured covariance matrix, the means, SEs, and Satterthwaite df are the same as from emmeans() on the same model, without a separate call. The means are returned as predMeans in the saved result (one data frame per factor) but are not created as separate objects by saveObjects.

Note: For any fixed effect using standard factor (treatment) coding, the Fixed Effects table also adds a row naming that predictor's reference category (e.g. group (ref = control)), placed immediately before its own rows.

What it saves automatically

When saveObjects=TRUE: Model0_Fixed (the fixed-effects table) and Model0_sigma2 (the residual variance). For a varIdent() model, also Model0_ResidByGroup (the by-group table) and Model0_sigma2_<group> for each group's own residual variance.

lmeSummary()

For multilevel models fit with nlme::lme()

Serves the same purpose as lmerSummary(), for a multilevel model fit with nlme's lme() function instead of lme4's lmer(). Supports the same ddf/ICC1/Neff/ICC2/slopeRel/randCI/randCIfixed/randCIrandom/LRT/effectSizes/CI options as lmerSummary(), except LRTdrop/LRTdropGroup—nlme's random-effects structure has no ranova()-style per-term breakdown for a custom row to fit into, so LRT here is a set of separate comparisons, described below: just one (the random effects) for a model with only a random intercept, and

otherwise one for each piece (each random slope, the set of random slopes, and the residual correlation). One difference under the hood: by default it replaces nlme's own degrees-of-freedom with a genuine Satterthwaite approximation (computed via the emmeans package) so the significance tests match what lmerSummary() reports for an equivalent lmer model. This is done for every fixed effect separatelythe intercept, continuous slopes, factor- or contrast-coded terms, and interactions alikeso each row of the Fixed Effects table has its own df. The overall test that all fixed slopes are zero uses the same Satterthwaite approximation, extended to several df at once, so it is an F-test even when the slopes are at different levels of the model (e.g. a twin-level and a pair-level predictor, where nlme's own anova(Terms=) cannot combine them); with ddf="asymptotic", or if the Satterthwaite df cannot be computed, it is a multivariate Wald chi-square test instead (labeled accordingly). In a no-intercept model with a factor entered as cell means (e.g. y ~ 0 + group + x), that factor's coefficients are the group means, not slopes, so they are left out of this test. For a model fit with weights=varIdent(form=~1|group) (separate residual variances by group, e.g. twins by zygosity), it also automatically prints a Residual Variance by Group table, and the ICC1/Neff/ICC2/slopeRel/randCI sections below automatically switch to a by-group breakdown instead of a single pooled value. For a model with a residual correlation structure (for example corAR1, corCompSymm, or corSymm), its parameter(s) are added to the Model for the Variance table as extra Residual rows in the Corr column (e.g., AR1 Phi), because nlme keeps them outside the random-effects variances; they are for display only and are not saved by saveObjects.

Basic usage

```
lmeSummary(Model2c, ICC2=TRUE, randCI=TRUE)
```

Arguments

Argument	Default	What it does
`model`	(required)	The fitted lme model object, entered without quotes (e.g. Model2c).
`name`	NULL	An optional text label (in quotes) printed above the output.
`ddf`	"Satterthwaite"	The method used for the fixed-effects tests' degrees of freedom: "Satterthwaite" (the recommended default) or "asymptotic" (treats df as infinite, i.e. a Wald z-test, and a Wald chi-square for the overall slope test). Kenward-Roger is not offered here (see note below). Effect sizes (below) are not available when ddf="asymptotic", since their formulas require a finite df.
`CI`	FALSE	Set to TRUE to add a confidence interval to the Fixed Effects table.
`levelCI`	95	The confidence level (%) used when CI=TRUE.
`predMeans`	FALSE	Set to TRUE to print the model-predicted mean of every level of each factor predictor, or give the name(s) of just the factor(s) you want in quotes, e.g. predMeans="time_F". One table per factor (Level, Mean, SE, df, plus Lower/Upper when CI=TRUE) prints right after the Fixed Effects table. It works whatever the factor's coding is (indicator, sequential, and so on)— see the note below.

Argument	Default	What it does
residPlot	FALSE	Set to TRUE to draw three plots of the level-1 residuals, side by side, for checking the normality and constant-variance assumptions: a histogram with a normal curve laid over it, a normal Q-Q plot (points close to the line = approximately normal), and the residuals against the fitted values with a zero line and a smoothed trend line (an even band around zero = constant variance; a funnel or curve = a problem). The residuals are the model's own conditional level-1 residuals, so the fitted values include the random effects. For a varIdent() model, whose residual spread is meant to differ by group, the residuals are standardized by the model's variance function first (the axes say "standardized"), so an even band means the modeled variances fit. For a model with a residual correlation structure (e.g. corAR1 or corSymm), the residuals are instead normalized (scaled for the variances and with the estimated within-cluster correlations removed; the axes say "normalized"), since raw residuals are correlated by design there; a model that fits then gives plots that look like independent standard normal noise. Instead of TRUE, give the name of a variable in the model's data in quotes, e.g. residPlot="time_F", to add a fourth plot of boxplots of the residuals at each level of that variable (similar box heights = constant variance across the levels) and to print the residuals' N, mean, and SD at each level; the variable does not have to be in the model. Drawn after all the printed output; nothing is added to the printed tables.
effectSizes	FALSE	Set to TRUE to print Cohen's d and partial r for each fixed slope (requires ddf="Satterthwaite").
ICC1	FALSE	Set to TRUE to print the ICC1 intraclass correlation. For a varIdent() model, prints one ICC1 per group instead of a single pooled value. Otherwise both an adjusted and an unadjusted ICC1 are printed (see lmerSummary() above), computed directly from the variance components, so they match lmerSummary() for the same model, including for nested grouping factors (performance::icc() gives wrong values for nested lme models). With nested grouping factors it also prints the proportion of variance by source and, for three levels, L3 / (L2 + L3), each adjusted and unadjusted (see the note below). A model with random slopes uses performance::icc().
Neff	FALSE	Set to TRUE to print the design effect (using the mean cluster size and ICC1) and the resulting effective sample size. For a varIdent() model, prints one design effect/effective N per group instead.
ICC2	FALSE	Set to TRUE to print the ICC2 reliability of the cluster means. For a varIdent() model, prints one ICC2 per group instead.
slopeRel	FALSE	Set to TRUE to print the slope reliability (SR) of each random slope, along with the slope variance, residual variance, predictor variance, and mean L1n it is computed from (see the glossary

Argument	Default	What it does
		above). Requires at least one random slope in the model. For a varIdent() model, prints one SR per slope per group instead.
`randCI`	FALSE	Set to TRUE to print confidence intervals for the random effects, automatically paired with the fixed effect of the same name. Automatically set to TRUE if randCIfixed and randCIrandom are both specified. Each row is labeled with the random effect and its grouping factor, "Term | Group" (for example, "Intercept | schoolID"), the same style as the LRT table. For a varIdent() model, also prints one intercept CI per group, built from that group's own fixed intercept (its mean when every other predictor is 0) and its own random intercept variance (the random effect named for the group, e.g. the MZ and DZ intercepts of a twin model, or a shared random intercept if there is only one); this works whether the group means are entered as cell means (y ~ 0 + group) or with treatment coding.
`randCIfixed / randCIrandom`	NULL	Use together to add one or more custom-paired CIs for a fixed effect and a random effect that do not share the same name—e.g. a level-2 predictor's CI on the random intercept. Enter the name of each predictor in quotes; pass equal-length vectors (e.g. randCIfixed=c("PaidvRed","UnpaidvRed"), randCIrandom=c("WC_PaidvRed","WC_UnpaidvRed")) to request more than one custom pairing at once, matched up by position. The intercept can be named Intercept, as the tables print it, or "(Intercept)".
`LRT`	FALSE	Set to TRUE to print likelihood ratio tests for the random effects. For a model with a random intercept only and no residual correlation structure, this is one test comparing the full model to a gls model with the same fixed effects and no random effects (a first row, "<none>", for the full model with nothing dropped, as in lmerSummary(), and one row, labeled like "Intercept | PersonID", for the random effects dropped). For a model with random slopes and/or a residual correlation structure (for example corAR1), each piece is instead tested separately, each against the full model: each random slope on its own (that slope's variance and its covariances with the other random effects are dropped, and any other slopes stay in the model); when there are two or more slopes, the set of all the random slopes together (the comparison model has a random intercept only); and the residual correlation structure (refit with none). Nothing is refit with a different estimation method: because the fixed effects are identical, every comparison model is fit with the same method the model itself used (REML by default), and for a varIdent() model it gets the same residual variance function, so only the piece being dropped is being tested. The tables first column is "Random Effect Dropped", matching every other *Summary() functions LRT table, and its first row, "<none>", is the full model with nothing dropped, as in lmerSummary(). The other rows are labeled the

Argument	Default	What it does
		same way (for example, "WPnm | PersonID" for a random slope). A p.mix column (the Self and Liang boundary-corrected p-value) is given for the tests that add one variance (a single random slope, or a lone random intercept) and is blank for the others, which are not simple boundary tests; use the p.value column for them.
`digits`	3	The number of decimal places shown in the printed tables.
`saveObjects`	FALSE	Set to TRUE to save the fixed-effects table and every variance/covariance component from the model as their own named objects (see “What it saves automatically” below). Off by default.
`envir`	(automatic)	Where the objects below are stored when saveObjects=TRUE. Leave at its default.

Note: Kenward-Roger DF is not available for lme models. emmeans's Kenward-Roger support requires the pbkrtest package, which only works for lmer models. Use lmerSummary() instead if you need Kenward-Roger df.

Note: For any fixed effect using standard factor (treatment) coding, the Fixed Effects table also adds a row naming that predictor's reference category (e.g. group (ref = control)), placed immediately before its own rows.

Note: nlme computes lme’s Satterthwaite df through emmeans using a simulation step, so the df are approximate (they can differ slightly from the same model fit in lmer) and would otherwise change a little from run to run; lmeSummary() runs that step with a fixed random seed, and restores your own random-number state afterward, so the df are reproducible and your own set.seed() sequence is not disturbed.

Note: For a model with nested grouping factors (e.g. random=~1|school/class), the Model for the Variance table lists every factor and the "groups" line shows each one's count. ICC1 additionally prints the proportion of variance by source and, for a three-level model, a second ICC, L3 / (L2 + L3), each adjusted and unadjusted; Neff prints the multi-level design effect (listing the mean n per level); ICC2 prints one value per level; and randCI labels each intercept with its factor, exactly as described for lmerSummary() above. slopeRel uses each slope's own factor's mean cluster size. A model that also has weights=varIdent() keeps the single-factor by-group output described above instead.

Note: With predMeans, each level's predicted mean is the Intercept plus that level's own coding, so it comes out right for indicator, 0/1 sequential, and centered sequential coding alike, exactly as described for lmerSummary() above (every other predictor is held at its coded zero point, and the table heading says "(other predictors at 0)" when the model has any). The means and SEs match lmerSummary() for the same model. The df are the same emmeans-based Satterthwaite df the Fixed Effects table uses (computed with the same fixed random seed, so they are reproducible); ddf="asymptotic" gives df = Inf. It works for nested random effects and for varIdent() models, since neither changes the fixed effects. The means are returned as predMeans in the saved result (one data frame per factor) but are not created as separate objects by saveObjects. A name that is not a factor predictor in the model stops with an error listing the model's factors.

What it saves automatically

When saveObjects=TRUE, the same naming scheme as lmerSummary(): Model2c_VarCorr, Model2c_Fixed, Model2c_tau0, Model2c_sigma2, and so on for each variance component. For a varIdent() model, also Model2c_ResidByGroup (the by-group residual variance table) and Model2c_sigma2_<group> for each group's own residual variance. With nested grouping factors, each variance/covariance name also gets its factor appended (e.g. Model2c_tau0_school and Model2c_tau0_class), and the saved variance table gains a grp column.

glmerSummary()

For generalized (binary/count outcome) models fit with lme4::glmer()

Parallel in structure to lmerSummary(), but for a model fit with glmer()—lme4's version for a binary, count, or other non-Gaussian outcome (e.g. a logistic multilevel model for a yes/no outcome). Several things that make sense for lmerSummary()'s Gaussian models do not translate directly to a GLMM, so glmerSummary() adapts them:

- **No Satterthwaite/Kenward-Roger df** glmer() fixed effects are tested with Wald z-tests, so there is no ddf argument here; p-values and CIs use the normal distribution.
- **Omnibus test** the joint test that all fixed slopes are zero is a multivariate Wald chi-square test (labeled accordingly) rather than an F-test, since glmer models have no residual-variance-based denominator df.
- **Effect sizes Cohen's d/partial r do not apply to a Wald z-test. Instead, for a logit-link model this prints the odds ratio (OR = exp(Estimate)) with its own delta-method SE (OR.SE = OR×SE) and, when CI=TRUE, a 95% CI—under the header "Effect Sizes for Fixed Slopes in Odds Ratios (OR)". For a log-link model (e.g. Poisson), the rate ratio (RR/RR.SE) instead, under the parallel "...in Rate Ratios (RR)" header. For any other link (e.g. probit), a note prints instead of a table, since an exponentiated effect size is not meaningful there. The CI comes from exponentiating the logit/log-scale CI's own endpoints, not from Estimate±1.96×SE, since that symmetric interval is not valid on a ratio scale.**
- **ICC2 / Neff (design effect) both require a level-1 ("residual") variance, which a GLMM does not estimate the way a Gaussian model does. These use the standard fixed constant instead—pi^2/3 for a logit link, 1 for a probit link (the same convention performance::icc() uses internally for ICC1), pi^2/6 for a cloglog link—shown as its own row in the Model for the Variance table, labeled with which constant applies and with its Variance/Std.Dev. in parentheses to flag it as assumed rather than estimated. Unavailable, with a clear note, for any other family/link.**
- **LRT delegates directly to glmerRanova() (see below), which already does exactly this for glmer models, including its "Random Effect Dropped" column and LRTdrop/LRTdropGroup custom-row option.**
- **Predicted probability (or mean) the Fixed Effects table also gets a Prob/Prob.SE column pair (plus Prob.Lower/Prob.Upper when CI=TRUE), converting the intercept's own linear-predictor-scale estimate to its actual predicted value via the model's own inverse link function, with a delta-method SE—blank for every slope row, since a slope has no single value of its own to convert. For a binomial model this is a genuine probability (bounded 0-**

1); for any other family (e.g. Poisson), family$linkinv() instead gives a predicted mean count/rate, so the columns are labeled Pred.Count/Pred.Count.SE there instead.

Basic usage

```
EmptyNon = glmer(data=Example3, family=binomial(link="logit"), nAGQ=7,
                 formula=nonwhite~1+(1|schoolID))
glmerSummary(EmptyNon, ICC1=TRUE, effectSizes=TRUE, CI=TRUE)
```

Arguments

Argument	Default	What it does
`model`	(required)	The fitted glmer model object, entered without quotes.
`name`	NULL	An optional text label (in quotes) printed above the output.
`CI`	FALSE	Set to TRUE to add a (normal-distribution) confidence interval to the Fixed Effects table.
`levelCI`	95	The confidence level (%) used when CI=TRUE.
`predMeans`	FALSE	Set to TRUE to print the model-predicted value for every level of each factor predictor, or give the name(s) of just the factor(s) you want in quotes, e.g. predMeans="condition". One table per factor prints right after the Fixed Effects table— see the note below for its columns.
`effectSizes`	FALSE	Set to TRUE to print the odds ratio (OR, logit link) or rate ratio (RR, log link) for each fixed slope, each with its own delta-method SE; a 95% CI is added too when CI=TRUE.
`ICC1`	FALSE	Set to TRUE to print the ICC1 intraclass correlation, both adjusted and unadjusted (the unadjusted one also adds the variance of the fixed effects' predictions to the denominator; see lmerSummary() above), computed from the assumed level-1 constant for a logit, probit, or cloglog link. Another family uses performance::icc().
`Neff`	FALSE	Set to TRUE to print the design effect and effective sample size (logit/probit links only).
`ICC2`	FALSE	Set to TRUE to print the ICC2 reliability of the cluster means (logit/probit links only).
`slopeRel`	FALSE	Set to TRUE to print the slope reliability (SR) of each random slope, using the same assumed level-1 constant as ICC2 for the residual variance (logit/probit links only). Requires at least one random slope in the model.
`randCI`	FALSE	Set to TRUE to print confidence intervals for the random effects, automatically paired with the fixed effect of the same name. Automatically set to TRUE if randCIfixed and randCIrandom are both specified. Each row is labeled with the random effect and its grouping factor, "Term | Group" (for example, "Intercept | schoolID"), the same style as the LRT table.

Argument	Default	What it does
`randCIfixed / randCIrandom`	NULL	Use together to add one or more custom-paired CIs for a fixed effect and a random effect that do not share the same name—e.g. a level-2 predictor's CI on the random intercept. Enter the name of each predictor in quotes; pass equal-length vectors (e.g. randCIfixed=c("PaidvRed","UnpaidvRed"), randCIrandom=c("WC_PaidvRed","WC_UnpaidvRed")) to request more than one custom pairing at once, matched up by position. The intercept can be named Intercept, as the tables print it, or "(Intercept)".
`LRT`	FALSE	Set to TRUE to print a likelihood ratio test for the random effect(s), via glmerRanova()--see that function below for what it tests and for the table's "Random Effect Dropped" column. Automatically set to TRUE if LRTdrop is specified.
`LRTdrop`	NULL	Add one extra row to the LRT table testing a custom, user-chosen combination of random effects removed together, all from ONE random-effects dimension -- see glmerRanova() below for the full description (it takes the identical argument).
`LRTdropGroup`	NULL	Only needed if the same random-effect name in LRTdrop occurs in more than one random-effects dimension -- see glmerRanova() below.
`digits`	3	The number of decimal places shown in the printed tables.
`saveObjects`	FALSE	Set to TRUE to save the fixed-effects table and every variance/covariance component from the model as their own named objects (see “What it saves automatically” below). Off by default.
`envir`	(automatic)	Where the objects below are stored when saveObjects=TRUE. Leave at its default.

Note: For any fixed effect using standard factor (treatment) coding, the Fixed Effects table also adds a row naming that predictor's reference category (e.g. group (ref = control)), placed immediately before its own rows.

Note: For a model with more than one random-intercept grouping factor (e.g. students in classrooms in schools), ICC1 additionally prints the proportion of variance by source (the assumed level-1 constant plays the residual's role) and, for a three-level nested model, a second ICC, L3 / (L2 + L3), each adjusted and unadjusted; Neff prints the multi-level design effect and ICC2 one value per level, as described for lmerSummary() above. Neff and ICC2 are "not available" for crossed factors, and, as before, for any family/link without a known level-1 constant. randCI shows one row per grouping factor's random intercept, labeled with that factor.

Note: With predMeans, each level's prediction is the Intercept plus that level's own coding (so it is right for indicator, 0/1 sequential, and centered sequential coding alike), with every other predictor at its coded zero point and the random effects at zero, as described for lmerSummary() above. Each table has the level, Est and SE on the model's link scale (as in the Fixed Effects table), then the same

prediction on the response scale with a delta-method SE: Prob/Prob.SE for a binomial model, or Pred.Count/Pred.Count.SE for other families such as Poisson. With CI=TRUE the Lower/Upper columns give the link-scale interval and Prob.Lower/Prob.Upper (or Pred.Count.Lower/Pred.Count.Upper) the same two endpoints transformed to the response scale. There is no df (Wald z). The means are returned as predMeans in the saved result but are not created as separate objects by saveObjects.

What it saves automatically

When saveObjects=TRUE, the same naming scheme as lmerSummary(): <model>_VarCorr, <model>_Fixed, <model>_tau0, and so on for each variance component, plus <model>_sigma2 for the assumed level-1 constant when the link supports one (logit, probit, or cloglog). With more than one grouping factor, each variance/covariance name also gets its factor appended (e.g. <model>_tau0_school, <model>_tau0_class).

clmmSummary()

For multilevel models with an ORDINAL outcome, fit with ordinal::clmm()

Parallel in structure to glmerSummary(), but for a model fit with clmm()—the ordinal package's cumulative link mixed model, the multilevel extension of an ordinal/proportional-odds logistic regression, for an outcome with 3 or more ordered categories (e.g. a Likert-type item). Like glmer models, clmm models are tested with Wald z-tests (no Satterthwaite/Kenward-Roger df) and have no true residual variance, so most of glmerSummary()'s same adaptations apply here too (see that entry above): a Wald chi-square omnibus test, an odds ratio (logit link only) for effect sizes, and an assumed level-1 variance constant standing in for ICC1/ICC2/Neff/slope reliability. The one real structural difference from glmer: an ordinal outcome with k categories has k-1 threshold (cutpoint) parameters instead of a single intercept. clmm() fits proportional-odds models only (it has no nominal= or scale= options and no test of proportional odds); for a model without random effects, or to relax proportional odds, use clmSummary() below.

- **Intercepts / Thresholds printed as two small tables, since the k-1 cutpoints replace the role an intercept plays elsewhere (both are excluded from the omnibus test, the Fixed Effects table, effect sizes, and reference-level detection, the same way an intercept already is everywhere else). Thresholds are clmm's own parameterization, predicting the chosen link of the probability of being in the LOWER category (at or below each cutpoint). Intercepts are the same cutpoints sign-flipped, predicting the probability of being in the HIGHER category—pointing the same direction as the slopes, which is why Intercepts prints first, immediately before Fixed Effects. Thresholds prints last, after Effect Sizes, and only when thresholds=TRUE (off by default, since Intercepts already carries the same information in the more intuitive direction). Both tables label each row with every category on each side of that cutpoint, e.g. a 3-category outcome's two cutpoints are labeled "0|1,2" and "0,1|2", and both add a Prob/Prob.SE column pair (plus Prob.Lower/Prob.Upper when CI=TRUE) converting that row's estimate to an actual probability via the model's own inverse link function, with a delta-method SE.**
- **Effect sizes the odds ratio (OR = exp(Estimate)) with its own delta-method SE (OR.SE), and a 95% CI when CI=TRUE, for each fixed slope—logit link only, since an exponentiated**

effect size is not meaningful for clmm's other four links, where a note prints instead. Header reads "Effect Sizes for Fixed Slopes in Odds Ratios (OR)" when shown.

- **Link scope the assumed level-1 variance constant covers all five links clmm() supports: pi^2/3 for logit, 1 for probit, pi^2/6 for cloglog or loglog, and unavailable for cauchit (the Cauchy distribution has no finite variance).**
- **LRT unlike glmerSummary() (which delegates to glmerRanova()), this is a ranova()-style table built directly for clmm models: one row per random-effects term, each refit with that whole term dropped (falling back to a plain clm() model when no random effects remain) and compared to the full model; a term that has random slopes gets one row per slope instead (that slope and its covariances dropped, the rest of the term kept) and no separate intercept row, as in lmerSummary()'s ranova() table, with the same p.mix mixture-corrected p-value used everywhere else in this guide. Each row's "Random Effect Dropped" label is condensed to "Term | Group" form, e.g. "(1 | schoolID)" is shown as "Intercept | schoolID". LRTdrop/LRTdropGroup add one extra custom row, the same as lmerSummary()'s (see that entry above).**

Basic usage

```
EmptyLunch = clmm(data=Example4b, link="logit",
formula=as.factor(lunch)~1+(1|schoolID))
clmmSummary(EmptyLunch, ICC1=TRUE, effectSizes=TRUE, CI=TRUE, thresholds=TRUE)
```

Arguments

Argument	Default	What it does
`model`	(required)	The fitted clmm model object, entered without quotes.
`name`	NULL	An optional text label (in quotes) printed above the output.
`CI`	FALSE	Set to TRUE to add a (normal-distribution) confidence interval to the Fixed Effects, Intercepts, and Thresholds tables, and to Effect Sizes.
`levelCI`	95	The confidence level (%) used when CI=TRUE.
`predMeans`	FALSE	Set to TRUE to print the model-predicted cumulative probabilities for every level of each factor predictor, or give the name(s) of just the factor(s) you want in quotes, e.g. predMeans="condition". One table per factor prints right after the Fixed Effects table— see the note below for its rows and columns.
`thresholds`	FALSE	Set to TRUE to also print the Thresholds table (see "Intercepts / Thresholds" above), after Effect Sizes. Off by default.
`effectSizes`	FALSE	Set to TRUE to print the odds ratio (OR) and its delta-method SE (OR.SE) for each fixed slope (logit link only); a 95% CI is added too when CI=TRUE.
`ICC1`	FALSE	Set to TRUE to print the ICC1 intraclass correlation, computed from the assumed level-1 variance constant (see the description above).

Argument	Default	What it does
`Neff`	FALSE	Set to TRUE to print the design effect and effective sample size (any link except cauchit).
`ICC2`	FALSE	Set to TRUE to print the ICC2 reliability of the cluster means (any link except cauchit).
`slopeRel`	FALSE	Set to TRUE to print the slope reliability (SR) of each random slope (any link except cauchit). Requires at least one random slope in the model.
`randCI`	FALSE	Set to TRUE to print confidence intervals for the random effects, automatically paired with the fixed effect of the same name. Since clmm has no single fixed Intercept the way lmer/glmer do, a random intercept term instead gets one CI per cutpoint's own Intercept estimate (all sharing the same random intercept variance), and its rows start with the cutpoint, e.g. "1|2,3,4,5 Intercept | judge". Automatically set to TRUE if randCIfixed and randCIrandom are both specified. Each row is labeled with the random effect and its grouping factor, "Term | Group" (for example, "Intercept | schoolID"), the same style as the LRT table.
`randCIfixed / randCIrandom`	NULL	Use together to add one or more custom-paired CIs for a fixed effect and a random effect that do not share the same name—e.g. a level-2 predictor's CI on the random intercept. Enter the name of each predictor in quotes; pass equal-length vectors (e.g. randCIfixed=c("PaidvRed","UnpaidvRed"), randCIrandom=c("WC_PaidvRed","WC_UnpaidvRed")) to request more than one custom pairing at once, matched up by position. The intercept can be named Intercept, as the tables print it, or "(Intercept)".
`LRT`	FALSE	Set to TRUE to print a likelihood ratio test for the random effect(s)--see "LRT" above for what it tests. Automatically set to TRUE if LRTdrop is specified.
`LRTdrop`	NULL	Add one extra row to the LRT table testing a custom, user-chosen combination of random effects removed together, all from ONE random-effects dimension -- see lmerSummary()'s own LRTdrop above for the full description (it takes the identical argument).
`LRTdropGroup`	NULL	Only needed if the same random-effect name in LRTdrop occurs in more than one random-effects dimension -- see lmerSummary() above.
`digits`	3	The number of decimal places shown in the printed tables.
`saveObjects`	FALSE	Set to TRUE to save the fixed-effects table and every variance/covariance component from the model as their own named objects (see “What it saves automatically” below). Off by default.

Argument	Default	What it does
`envir`	(automatic)	Where the objects below are stored when saveObjects=TRUE. Leave at its default.

Note: For any fixed effect using standard factor (treatment) coding, the Fixed Effects table also adds a row naming that predictor's reference category (e.g. group (ref = control)), placed immediately before its own rows. If the model has no fixed slopes at all, the Fixed Effects table still prints its column headers, with a single "(none)" row in place of any real rows.

Note: For a model with more than one random-intercept grouping factor, ICC1 is the share of the total variance from ALL the random intercepts together (their sum over that sum plus the assumed constant), and additionally prints the proportion of variance by source and, for a three-level nested model, a second ICC, L3 / (L2 + L3), each adjusted and unadjusted (the unadjusted ones add the variance of the fixed effects' predictions to the denominator); Neff prints the multi-level design effect and ICC2 one value per level, as described for lmerSummary() above. Neff and ICC2 are "not available" for crossed factors, and for the cauchit link. randCI shows each cutpoint's row once per grouping factor's random intercept, labeled with that factor.

Note: A clmm has no single intercept, so with predMeans each factor level gets one row per cutpoint, labeled the same way as the Intercepts table (e.g. "0|1,2"). The Est is the link-scale value for the probability of being in the HIGHER category at that cutpoint— the cutpoint's (sign-flipped) threshold plus that level's own coding— pointing the same direction as the Intercepts table and the slopes, with Prob/Prob.SE converting it to a probability (delta-method SE). So at the coded zero point (e.g. a treatment-coded factor's reference level) the rows equal the Intercepts table exactly. Every other predictor is held at its coded zero point and the random effects at zero; the coding can be indicator, 0/1 sequential, or centered sequential. With CI=TRUE, Lower/Upper give the link-scale interval and Prob.Lower/Prob.Upper the same endpoints as probabilities. The means are returned as predMeans in the saved result but are not created as separate objects by saveObjects.

```
What it saves automatically
```

When saveObjects=TRUE, the same naming scheme as glmerSummary(): <model>_VarCorr, <model>_Fixed, <model>_Thresholds, <model>_Intercepts, <model>_tau0, and so on for each variance component, plus <model>_sigma2 for the assumed level-1 constant when the link supports one (logit, probit, cloglog, or loglog). With more than one grouping factor, each variance/covariance name also gets its factor appended (e.g. <model>_tau0_school, <model>_tau0_class).

clmSummary()

For single-level models with an ORDINAL outcome, fit with ordinal::clm(), including non-proportional-odds effects

clmmSummary()'s counterpart for an ordinal outcome modeled without random effects, with clm()— the ordinal package's cumulative link (proportional-odds) model, for an outcome with 3 or more ordered categories. It prints the same cutpoint and slope tables as clmmSummary(): an Intercepts table (the k-1 cutpoints sign-flipped, giving the link of the probability of being in the HIGHER category after each cutpoint, so they point the same direction as the slopes), the Fixed Effects table with Wald z-tests, an overall Wald chi-square test that all slopes are zero, and optionally odds ratios

(logit link only), model-predicted probabilities, and a Thresholds table. It has no variance, ICC, or random-effect output, since there are no random effects.

It also reports what only clm() can do. A predictor given in clm()'s nominal= argument has its own effect at every cutpoint instead of one slope (non-proportional odds); these get a Nominal Effects table, one row per predictor and cutpoint, signed like the slopes (positive = more likely above that cutpoint). A predictor given in scale= changes the spread of the latent variable; these get a Scale Effects table on the log scale, with SD.Ratio = exp(Est), the ratio of the latent SD per unit of the predictor. The options LRT, R2, and parallelTest add likelihood ratio tests, pseudo-R2, and the test of proportional odds (see below). Only clm()'s default threshold="flexible" and the five links logit, probit, cloglog, loglog, and cauchit are supported. (clmm() has no nominal= or scale= options and no test of proportional odds, so these are available only here.)

Basic usage

From the wine data in the ordinal package (ratings of 1 to 5 by temperature and contact). Model2 gives contact its own effect at each cutpoint.

```
Model1 = clm(rating ~ temp + contact, data=wine)
clmSummary(Model1)
clmSummary(Model1, CI=TRUE, effectSizes=TRUE, predMeans="temp",
           LRT=TRUE, R2=TRUE, parallelTest=TRUE)
Model2 = clm(rating ~ temp, nominal=~contact, data=wine)
clmSummary(Model2)
```

Output of the first call:

```
Model Fit for Predicting rating:
      AIC       BIC    logLik -2*logLik
  184.984   198.644   -86.492   172.984
Number of obs: 72

Overall Wald Chi-Square Test for All Fixed Slopes:
 Model  NumDF  Chisq p.value
 Model1     2 27.912   <.001

Intercepts predicting the logit of the probability
of being in the HIGHER category after | :
 Term          Est    SE z.value p.value  Prob Prob.SE
 1|2,3,4,5   1.344 0.517   2.600   0.009 0.793   0.085
 1,2|3,4,5  -1.251 0.438  -2.857   0.004 0.223   0.076
 1,2,3|4,5  -3.467 0.598  -5.800   <.001 0.030   0.018
 1,2,3,4|5  -5.006 0.731  -6.850   <.001 0.007   0.005

Fixed Effects:
 Term                  Est    SE z.value p.value
 temp (ref = cold)
 tempwarm            2.503 0.529   4.735   <.001
 contact (ref = no)
 contactyes          1.528 0.477   3.205   0.001
```

Some of the extra tables: the next three come from LRT=TRUE, R2=TRUE, and parallelTest=TRUE for Model1, and the last is the Nominal Effects table printed for Model2:

```
LRT for each fixed slope (full model vs. that term dropped):
 Term Dropped  Npar   LL -2LL AIC BIC    LRT DF p.value
 <none>           6  -86  173 185 199
 temp             5 -100  200 210 221 26.928  1   <.001
 contact          5  -92  184 194 205 11.043  1   0.001
 All slopes       4 -104  207 215 225 34.454  2   <.001

Pseudo-R2 (relative to a model with only the cutpoints):
 McFadden Cox.Snell Nagelkerke
    0.166     0.380      0.403

Test of Proportional Odds (a separate effect at each cutpoint):
 Term    DF   LRT p.value
 temp     3 3.175   0.365
 contact  3 0.567   0.904
  (significant = that predictor's effect differs across cutpoints)

Nominal Effects (a separate effect at each cutpoint):
 Term       Cutpoint     Est    SE z.value p.value
 contactyes 1|2,3,4,5  1.615 1.162   1.390   0.164
 contactyes 1,2|3,4,5  1.512 0.591   2.559   0.010
 contactyes 1,2,3|4,5  1.675 0.649   2.581   0.010
 contactyes 1,2,3,4|5  1.051 0.897   1.172   0.241
```

Arguments

Argument	Default	What it does
`model`	(required)	The fitted clm model object, entered without quotes.
`name`	NULL	An optional text label (in quotes) printed above the output.
`effectSizes`	FALSE	Set to TRUE to print the odds ratio (OR) and its delta-method SE (OR.SE) for each fixed slope (logit link only); a 95% CI is added too when CI=TRUE.
`CI`	FALSE	Set to TRUE to add a (normal-distribution) confidence interval to the Fixed Effects, Intercepts, Nominal Effects, Scale Effects, and Thresholds tables, and to Effect Sizes.
`thresholds`	FALSE	Set to TRUE to also print the Thresholds table (the cutpoints in clm()'s own direction, the link of the probability of being at or below each cutpoint), after Effect Sizes. Off by default.
`levelCI`	95	The confidence level (%) used when CI=TRUE.
`predMeans`	FALSE	Set to TRUE to print the model-predicted probability of being above each cutpoint for every level of each factor predictor (other predictors at 0), or give the name(s) of just the factor(s) you want in quotes, e.g. predMeans="temp". One table per factor prints

Argument	Default	What it does
		right after the Fixed Effects table. Not available for a model with nominal= or scale= terms.
`LRT`	FALSE	Set to TRUE to print a likelihood ratio test for each fixed slope (the full model against that term dropped, as drop1() does it, so interactions are respected) and for all the slopes together against the model without them.
`R2`	FALSE	Set to TRUE to print McFadden, Cox-Snell, and Nagelkerke pseudo-R2, each relative to a model with only the cutpoints.
`parallelTest`	FALSE	Set to TRUE to print the test of proportional odds, from ordinal::nominal_test(): for each predictor in the formula, a likelihood ratio test of letting its effect differ across cutpoints. A significant result says that predictor's proportional-odds assumption does not hold; consider moving it to nominal=.
`digits`	3	The number of decimal places shown in the printed tables.
`saveObjects`	FALSE	Set to TRUE to save the fixed-effects table (<model>_Fixed), the cutpoints (<model>_Thresholds and <model>_Intercepts), and, when the model has them, the nominal and scale effects (<model>_Nominal and <model>_Scale) as their own named objects. Off by default.
`envir`	(automatic)	Where the objects above are stored when saveObjects=TRUE. Leave at its default.

The results are also returned, invisibly, as a list (Fixed, Thresholds, Intercepts, Nominal, Scale, omnibusWald, effectSizes, fixedCI, predMeans, LRT, R2, and parallelTest, for whichever were requested or apply), e.g. x = clmSummary(Model1).

Note: clmSummary() is for a single-level model. For clustered ordinal data use clmmSummary(), which fits proportional-odds models only: clmm() ignores nominal= and scale= (clmmSummary() stops with a message if the model call has either) and has no test of proportional odds.

con1DSummary()

Formats a single custom contrast test

Formats the result of a single linear-combination ("1 df contrast") test—for example, testing whether one group's slope equals another's, or whether a specific combination of fixed effects equals zero—computed first with lmerTest's contest1D() function. It prints the result in the same table layout as the Fixed Effects table in lmerSummary(), so it looks consistent with the rest of your output.

Basic usage

The examples for con1DSummary(), con1DTable(), and conMDSummary() all use the same model, Model4c from Example 4a: math ~ 1 + WCses + WCnon + CMses + CMnon30 + (1|schoolID). Each level-1 predictor enters as a cluster-mean-centered within-cluster slope (WCses, WCnon) and a cluster-mean between-cluster slope (CMses, CMnon30), so its fixed effects, in order, are Intercept,

WCses, WCnon, CMses, and CMnon30. Its contextual effects (between minus within) are the contrasts tested below.

```
Test1 = contest1D(Model4c, L=c(0, -1, 0, 1, 0))    # CMses - WCses
con1DSummary(Test1, name="Contextual SES", CI=TRUE, effectSizes=TRUE)
```

```
Fixed Effect: Contextual SES (with 95% CI)
   Est    SE      DF t.value p.value Lower Upper
 3.375 0.417 172.761   8.090   <.001 2.551 4.198

Effect Size:
   Est    SE p.value Cohen.d Partial.r
 3.375 0.417   <.001   1.231     0.524
```

Arguments

Argument	Default	What it does
`term`	(required)	The saved result of a contest1D() call, entered without quotes—not the model itself.
`name`	NULL	An optional text label (in quotes), appended to the "Fixed Effect" table's own title (e.g. "Fixed Effect: SES slope").
`CI`	FALSE	Set to TRUE to add a confidence interval.
`levelCI`	95	The confidence level (%) used when CI=TRUE.
`effectSizes`	FALSE	Set to TRUE to print a separate Cohen's d / partial r effect-size table.
`digits`	3	The number of decimal places shown in the printed tables.

Note: since con1DSummary() only formats an already-computed contest1D() result, it cannot control which ddf method was used to compute it. If you called lmerSummary() with ddf="Kenward-Roger", pass ddf="Kenward-Roger" to your own contest1D() call too, for consistency (contest1D() has no "asymptotic" option).

Example: contextual effects from a within-between model (Example 4a)

In a model that has each predictor as a within-cluster (cluster-mean-centered) slope and a cluster-mean slope, the cluster-mean slope is the between-cluster effect, and the contextual effect is the difference between the two: between minus within. Each number in L goes with one fixed effect, in the order of the Fixed Effects table (Example 4a's Model 4c has Intercept, WCses, WCnon, CMses, CMnon30), so the contextual effect of SES, CMses − WCses, is c(0, -1, 0, 1, 0). With the fixed effects of Model 4c (WCses = 1.952 and CMses = 5.327) that is 5.327 − 1.952 = 3.375, along with its SE and test:

```
# Model 4c: contextual effect of SES = CMses - WCses
Test1 = contest1D(Model4c, L=c(0, -1, 0, 1, 0))
con1DSummary(Test1, name="Contextual SES")
Fixed Effect: Contextual SES
   Est    SE      DF t.value p.value
```

```
 3.375 0.417 172.761   8.090   <.001
```

contest1D() accepts L only by position, as above; con1DTable() (next) accepts the same contrast written by name, and can request several at once.

con1DTable()

Multiple contrast tests against the same model, as one combined table

Runs several single-df contrasts (tests of one linear combination of the fixed effects each) against the same model, an lmer, lme, or gls model, and prints them as a single table—one row per contrast—instead of one contest1D()/con1DSummary() call and printed block per contrast. Unlike con1DSummary(), it does the calculation for you, so you do not need to run anything first: for an lmer model it uses lmerTest's contest1D() (which works for lmer models only), and for an lme or gls model, which contest1D() cannot take, it computes each contrast directly from the model's fixed effects and covariance matrix, with the same Satterthwaite df method that lmeSummary() uses, and gives the same columns. To test more than one contrast at once (2 or more df), use conMDSummary() instead.

Basic usage

The examples for con1DSummary(), con1DTable(), and conMDSummary() all use the same model, Model4c from Example 4a: math ~ 1 + WCses + WCnon + CMses + CMnon30 + (1|schoolID). Each level-1 predictor enters as a cluster-mean-centered within-cluster slope (WCses, WCnon) and a cluster-mean between-cluster slope (CMses, CMnon30), so its fixed effects, in order, are Intercept, WCses, WCnon, CMses, and CMnon30. Its contextual effects (between minus within) are the contrasts tested below.

```
con1DTable(Model4c, L=list("Contextual SES"=c(WCses=-1, CMses=1),
                           "Contextual Nonwhite"=c(WCnon=-1, CMnon30=1)),
           CI=TRUE)

Requested Fixed Effects (with 95% CI):
 Name                  Est    SE      DF t.value p.value Lower Upper
 Contextual SES      3.375 0.417 172.761   8.090   <.001 2.551 4.198
 Contextual Nonwhite 1.351 0.594 199.136   2.274   0.024 0.179 2.523
```

Arguments

Argument	Default	What it does
`model`	(required)	The fitted lmer, lme, or gls model object, entered without quotes.
`L`	(required)	A named list of contrast vectors—one per row you want in the table. Each element's name (in quotes) becomes that row's label in the Name column, e.g. list("Contextual SES"=c(0,-1,0,1,0), "Contextual Nonwhite"=c(0,0,-1,0,1)). Every element must be named. Each contrast can be a full-length vector (one value per fixed effect, in model order) or, usually easier, a vector named after the fixed effects as they are printed in the Fixed Effects table, with every effect you do not name set to 0, e.g. list("Contextual SES"=c(WCses=-1, CMses=1), "Contextual Nonwhite"=c(WCnon=-1, CMnon30=1)). The intercept can be named Intercept, as the

Argument	Default	What it does
		Fixed Effects table prints it, e.g. c(Intercept=1, sexMW=1) ("(Intercept)", the name lme4 and nlme use, also works).
`ddf`	"Satterthwaite"	The method used for each contrast's degrees of freedom: "Satterthwaite" (the recommended default) or "Kenward-Roger" (lmer models only), passed through to contest1D() for an lmer model. For an lme or gls model (which does not use contest1D()), "Satterthwaite" is the same emmeans-based df that lmeSummary() uses (computed with a fixed random seed, so it is reproducible), or "asymptotic" gives df = Inf, a Wald z-test.
`CI`	FALSE	Set to TRUE to add Lower/Upper confidence-interval columns to the table.
`levelCI`	95	The confidence level (%) used when CI=TRUE.
`effectSizes`	FALSE	Set to TRUE to add Cohen.d and Partial.r columns to the same table.
`digits`	3	The number of decimal places shown in the printed table.

The printed table (headed "Requested Fixed Effects") has one row per contrast, with columns Name, Est, SE, df, t.value, p.value, plus Lower/Upper (if CI=TRUE) and Cohen.d/Partial.r (if effectSizes=TRUE)—everything con1DSummary() would print for one contrast, combined onto a single row per contrast instead of separate blocks.

Examples: contextual effects, written by position or by name (Example 4a)

The contextual effect of SES can be written by position, c(0, -1, 0, 1, 0), or by name, c(WCses=-1, CMses=1), with every effect you do not name set to 0. A named vector does not depend on the order of the effects and is easier to check. The first two rows below are the same contrast; the third is the contextual effect of the nonwhite predictor, written by name:

```
con1DTable(Model4c, L=list("Contextual SES (by position)"  = c(0, -1, 0, 1, 0),
                           "Contextual SES (by name)"      = c(WCses=-1,
CMses=1),
                           "Contextual Nonwhite (by name)" = c(WCnon=-1,
CMnon30=1)))
Requested Fixed Effects:
 Name                            Est    SE      DF t.value p.value
 Contextual SES (by position)  3.375 0.417 172.761   8.090   <.001
 Contextual SES (by name)      3.375 0.417 172.761   8.090   <.001
 Contextual Nonwhite (by name) 1.351 0.594 199.136   2.274   0.024
```

The same call works unchanged for the same model fit with lme() (lme(math ~ 1 + WCses + WCnon + CMses + CMnon30, random = ~ 1 | schoolID)); the estimates and SEs are identical and only the df differ slightly (174.787 and 201.468), since nlme's variance estimates are approximate.

If the level-1 predictors were instead grand-mean-centered (rather than cluster-mean-centered), the cluster-mean slope would already be the contextual effect, and no contrast would be needed.

emmSat()

emmeans() with Satterthwaite degrees of freedom set for you

A thin wrapper for emmeans::emmeans(), for getting model-predicted means or pairwise comparisons directly from an lmer, lme, or gls model. emmeans' own default denominator df differ by model type and cannot be changed with an option: containment df for an lme model, and Kenward-Roger df (if the pbkrtest package is installed) for an lmer model. emmSat() sets the df method for you, so the means, standard errors, and tests match the df that lmerSummary(), lmeSummary(), glsSummary(), con1DTable(), and conMDSummary() report. Everything else is called exactly as in emmeans().

Basic usage

The first call prints the model-predicted mean at each occasion (from Example 5b, a gls model with an unstructured R matrix); the second adds all pairwise differences between occasions, unadjusted; the third gives predicted means at chosen values of a numeric predictor (here from an lme model with random linear change):

```
emmSat(Model2b, ~occasion)
emmSat(Model2b, pairwise~occasion, adjust="none")
emmSat(Model3b, ~time, at=list(time=c(0, 3)))
```

Output of the first call (the means), and of the second call (which prints the same means under "emmeans:" and then the pairwise differences under "contrasts:"):

```
 occasion emmean    SE      df lower.CL upper.CL
 1         8.225 0.237 206.548    7.758    8.692
 2         9.080 0.230 194.243    8.626    9.534
 3         9.950 0.239 199.421    9.479   10.421
 4        11.070 0.271 200.661   10.536   11.604
Degrees-of-freedom method: appx-satterthwaite
Confidence level used: 0.95

contrasts:
 contrast              estimate    SE      df t.ratio p.value
 occasion1 - occasion2   -0.855 0.207 199.035  -4.126   <.001
 occasion1 - occasion3   -1.725 0.239 199.128  -7.230   <.001
 occasion1 - occasion4   -2.845 0.277 198.292 -10.257   <.001
 occasion2 - occasion3   -0.870 0.222 197.915  -3.922   <.001
 occasion2 - occasion4   -1.990 0.256 197.932  -7.765   <.001
 occasion3 - occasion4   -1.120 0.225 198.138  -4.973   <.001
Degrees-of-freedom method: appx-satterthwaite
```

Arguments

Argument	Default	What it does
`model`	(required)	The fitted lmer, lme, or gls model object, entered without quotes.
`specs`	(required)	What to compute means for, exactly as in emmeans(): a formula such as ~occasion (the mean at each level), ~time (at values given

Argument	Default	What it does
		with at=), or pairwise~occasion (the means plus all pairwise differences), or a character vector of names.
`ddf`	"Satterthwaite"	The denominator df method: "Satterthwaite" (the default; for lme and gls these are approximate df computed by simulation inside emmeans) or "asymptotic" (df = Inf, z-tests).
`digits`	3	The number of decimal places shown in the printed tables, as in the *Summary() functions; p-values are shown to at least 3 decimal places, as "<.001" when smaller. The labels in the first column(s) (a factor's levels, or the values given with at=) are shown as they are.
`...`		Any other emmeans() argument, passed through unchanged, e.g. at= (values of other predictors), adjust="none" (or "bonferroni", "tukey", etc.) for the comparisons, by=, or level= for the confidence level.

Note: Satterthwaite df for an lme or gls model are computed by simulation inside emmeans, so emmSat() uses a fixed random seed (and puts your own random-number state back afterward), which makes the df reproducible from run to run and the same as in this file's other functions.

The tables are always printed, in the same layout and with the same number formatting as the *Summary() functions, followed by emmeans' own notes about the df method, the confidence level, and any p-value adjustment; a pairwise~ specification prints two tables, labeled "emmeans:" and "contrasts:". The usual emmeans object is returned invisibly, so assign it (e.g. Means = emmSat(Model2b, ~occasion)) if you want to use summary(), confint(), pairs(), contrast(), or test() on it as with emmeans(). For a plain linear combination of fixed effects with Satterthwaite df, con1DTable() is an alternative; for a joint test, use conMDSummary().

conMDSummary()

Joint test of a chosen subset of a model's slopes, or of any linear combinations of its fixed effects

Tests a set of a model's fixed effects jointly, in one of two ways. With terms, it tests whether a chosen subset of the fixed slopes are all zero—for example, just the two SES-related slopes in a model that also has other predictors—using the same mechanism each *Summary() function's own "Overall F-test/Wald test for All Fixed Slopes" already uses for the whole model, just restricted to the slopes named in terms. With L, it tests any set of linear combinations of the fixed effects equal to zero—for example, that a within-cluster slope and a between-cluster slope are equal, that two group means differ, or several such questions at once. Works with a model fit by lmer(), lme(), gls(), or glmer(): lmerTest's contestMD() for an lmer model (Satterthwaite or Kenward-Roger df); for an lme model, an F-test with Satterthwaite denominator df (the same per-coefficient df lmeSummary()'s Fixed Effects table uses, extended to several df at once the way contestMD() does it) or, with ddf="asymptotic", a multivariate Wald chi-square test; for a gls model, the same Satterthwaite F-test used for lme (which accounts for an estimated residual covariance matrix such as an unstructured R matrix) or, with ddf="residual", an F-test on its residual df, or, with ddf="asymptotic", the Wald chi-square; and always a multivariate Wald chi-square test for a glmer model (since neither contestMD() nor anova(Terms=) supports glmer). Because it does not depend on anova(Terms=), it can test slopes at

different levels of a model together (e.g. a twin-level and a pair-level predictor), which nlme's own anova() cannot. Passing every one of a model's slope names reproduces the same result as that model's own *Summary() omnibus test, since that is just the special case of testing all of them at once.

Basic usage

The examples for con1DSummary(), con1DTable(), and conMDSummary() all use the same model, Model4c from Example 4a: math ~ 1 + WCses + WCnon + CMses + CMnon30 + (1|schoolID). Each level-1 predictor enters as a cluster-mean-centered within-cluster slope (WCses, WCnon) and a cluster-mean between-cluster slope (CMses, CMnon30), so its fixed effects, in order, are Intercept, WCses, WCnon, CMses, and CMnon30. Its contextual effects (between minus within) are the contrasts tested below.

```
conMDSummary(Model4c, terms=c("WCses","CMses"),
             name="Joint Test: SES (Within + Between)")
conMDSummary(Model4c, L=c(WCses=1, CMses=-1), name="Within = Between SES
slope")
conMDSummary(Model4c, L=list("Contextual SES"=c(WCses=-1, CMses=1),
                             "Contextual Nonwhite"=c(WCnon=-1, CMnon30=1)),
             name="Both contextual effects zero")
conMDSummary(Model4c, L=list("Contextual SES"=c(WCses=-1, CMses=1), "CMnon30"),
             name="Contextual SES and CMnon30 slope")
```

```
Joint Test: SES (Within + Between)
Joint Test for:
WCses, CMses
 Model   NumDF   DenDF F.value p.value
 Model4c     2 291.926 248.279   <.001

Within = Between SES slope
Joint Test of 1 Linear Combination of the Fixed Effects
 Model   NumDF   DenDF F.value p.value
 Model4c     1 172.761  65.453   <.001

Both contextual effects zero
Joint Test of 2 Linear Combinations of the Fixed Effects:
Contextual SES, Contextual Nonwhite
 Model   NumDF   DenDF F.value p.value
 Model4c     2 193.242  33.816   <.001

Contextual SES and CMnon30 slope
Joint Test of 2 Linear Combinations of the Fixed Effects:
Contextual SES, CMnon30
 Model   NumDF   DenDF F.value p.value
 Model4c     2 163.541  59.790   <.001
```

The first test asks whether SES matters at all (the within and between slopes both zero). The second, a single combination, tests that the within and between SES slopes are equal; this is the same question as the contextual effect of SES being zero, so its F (65.453) is the square of that contrast's t (8.090) in con1DSummary() and con1DTable(), with the same denominator df (172.761). The third

tests both contextual effects at once (2 df), and the fourth shows how a combination and a plain fixed-effect name can be listed together. The same L works unchanged for the lme() version of this model (for example, the two contextual effects give F = 33.816 with denominator df 195.506).

Arguments

Argument	Default	What it does
`model`	(required)	The fitted lmer, lme, gls, or glmer model object, entered without quotes.
`terms`	NULL	A character vector of fixed-effect term names to test jointly, e.g. c("WCses","CMses")—must exactly match names already in the model. The intercept cannot be included (use L to test it). Give exactly one of terms or L.
`L`	NULL	Any set of linear combinations of the fixed effects to test equal to zero, in one of these forms: a character vector of fixed-effect names (each tested = 0); a numeric vector for one combination, either with one value per fixed effect in model order or, more simply, named after the fixed effects with all unnamed effects counted as 0 (e.g. c(WCses=1, CMses=-1) for "the two slopes are equal"); a list mixing those forms, with one row per numeric vector (a list name labels it in the output) and one row per fixed-effect name in any character element, so a difference can simply be followed by plain names (e.g. list("Contextual SES"=c(WCses=-1, CMses=1), "CMnon30")); or a numeric matrix with one column per fixed effect. The intercept can be named Intercept, as the Fixed Effects table prints it ("(Intercept)", the name lme4 and nlme use, also works). A name that is not a fixed effect in the model stops with an error listing the valid names.
`name`	NULL	An optional text label (in quotes) printed above the output.
`ddf`	"Satterthwaite"	The method used for the test's degrees of freedom. For an lmer model: "Satterthwaite" (the recommended default) or "Kenward-Roger", passed to contestMD(). For an lme model: "Satterthwaite" (the Satterthwaite F-test described above) or "asymptotic" (a Wald chi-square with no denominator df). For a gls model: "Satterthwaite" (the default, as for lme), which uses each test's own Satterthwaite denominator df and is preferred for a gls model with an estimated R matrix; "residual", an F-test using the residual df N minus the number of fixed effects; or "asymptotic" (a Wald chi-square with no denominator df). Silently has no effect for a glmer model, which always uses the Wald chi-square.
`digits`	3	The number of decimal places shown in the printed table.

Note: Satterthwaite df for an lme or gls model are approximate (they can differ slightly from the same model fit in lmer) and are computed with a fixed random seed, so they are reproducible and your own random-number sequence is left undisturbed.

The printed output starts with a header line saying what was tested ("Joint Test for:" with terms, or how many linear combinations with L), followed on its own line by the list of terms (or the combinations' labels, if L has any), then a single-row table: Model, NumDF, DenDF, F.value, p.value for an F-test, or Model, NumDF, Chisq, p.value for a Wald chi-square test (which has no DenDF), followed by a short note whenever a Wald chi-square was used instead of an F-test (and, for a gls model, a note stating whether Satterthwaite or residual denominator df were used). A typo'd or invalid term name, or a term not found in the model, stops with a plain-English error listing the model's valid term names.

PseudoR2()

Compares variance components across a chain of models

Computes pseudo-R2 for each variance component (e.g., the random intercept variance, a random slope variance, the residual variance)—the percent reduction in that variance when moving from a baseline model to one or more larger models with more predictors. This shows how much of each source of variability your predictors explain. Up to four alternative models can be compared at once, e.g., baseline → add level-1 predictors → add level-2 predictors → add a cross-level interaction—with the change in pseudo-R2 shown at each additional step. Works with models fit by lmer() or by lme() (every model in one call must be the same type, not a mix). For an lme model with weights=varIdent() (separate residual variances by group), the residual gets one row per group, labeled "Residual (<group>)", each compared to the same group in the baseline model.

Basic usage

```
PseudoR2(baseModel=Model1, altModel1=Model2, altModel2=Model3)
```

For a model with more than one grouping factor (crossed random effects, as in Example 6a, or three or more nested levels), each term is labeled with its grouping factor, so that rows with the same name, such as the two random intercepts, can be told apart. Nested factors are listed from the highest level down.

```
PseudoR2(baseModel=Model1b, altModel1=Model2)

Pseudo-R2
 Term                    Model   Variance Pseudo.R2
 Intercept | PschoolID   Model1b    0.175
 Intercept | PschoolID   Model2     0.172     0.015
 Intercept | SschoolID   Model1b    0.079
 Intercept | SschoolID   Model2     0.073     0.079
 Residual                Model1b    0.505
 Residual                Model2     0.504     0.002
```

Arguments

Argument	Default	What it does
`baseModel`	(required)	The simplest ("empty"/unconditional) model, used as the comparison point that every alt model is measured against.
`altModel1`	(required)	The first, more complex model to compare against baseModel.

Argument	Default	What it does
`altModel2 / altModel3 / altModel4`	NULL	Optional additional models, each one compared to the model listed just before it. Must be supplied in order—you cannot skip altModel2 and supply only altModel3.
`digits`	3	The number of decimal places shown in the printed table.

Important: Every model being compared must have exactly the same random-effects structure (the same random intercepts/slopes) as baseModel—only the fixed effects may differ between models. If this is not the case, the function stops with an explanatory error, since pseudo-R2 is not meaningful when the random-effects structures differ across models. For lme models this includes the residual variance structure: if the baseline model has weights=varIdent(), so must every other model, with the same groups.

TotalR2()

Variance explained by the fixed effects only

Computes total-R2—the proportion of the total outcome variance explained by a model's fixed effects only—for a single model, or the change in total-R2 across up to four models with progressively more fixed effects. Works with models fit by lmer() or lme().

Basic usage

```
TotalR2(Model2, Model3, Model4)
```

Arguments

Argument	Default	What it does
`altModel1`	(required)	The first model to report Total-R2 for.
`altModel2 / altModel3 / altModel4`	NULL	Optional additional, more complex models, compared to the model listed just before each one. Must be supplied in order, as in PseudoR2().
`digits`	3	The number of decimal places shown in the printed table.

Note: when only altModel1 is supplied, the Change.R2 column is left out of the printed table entirely (there is no previous model to compare it to), rather than showing a blank column.

randLRT()

Likelihood ratio test between two separately-fit nested models

Computes a likelihood ratio test (LRT) between two models you have already fit separately, when they cannot be compared with ranova() or the LRT= option in lmerSummary()/lmeSummary()—most commonly, a single-level model (glm or gls, with no random effects) versus a nested multilevel model (glmer, lmer, or lme) that adds one or more random effects. This is the manual comparison used for GLMMs (binary/count outcomes), such as glm vs. glmer for a logistic model. Unlike a fixed

df=1 test, randLRT() computes the test's degrees of freedom automatically as the difference in the number of estimated parameters between the two models, so it works correctly even if the more complex model adds more than one random-effect parameter at once. The output table includes both the ordinary p-value and the p.mix mixture-corrected p-value (see the glossary above), which is the more appropriate one when fullModel adds a variance component to baseModel.

Basic usage

```
SingleNon = glm(data=Example3, family=binomial(link="logit"),
formula=nonwhite~1)
EmptyNon = glmer(data=Example3, family=binomial(link="logit"), nAGQ=7,
                 formula=nonwhite~1+(1|schoolID))
randLRT(SingleNon, EmptyNon, name="LRT for Random Intercept (Nonwhite)")
```

Arguments

Argument	Default	What it does
`baseModel`	(required)	The simpler model (e.g. a glm or gls with no random effects), entered without quotes.
`fullModel`	(required)	The more complex, nested model (e.g. a glmer, lmer, or lme model that adds random effects to baseModel), entered without quotes.
`name`	NULL	An optional text label (in quotes) printed above the output.
`digits`	3	The number of decimal places shown in the printed table (the LRT statistic and p-value; LL/-2LL/AIC/BIC are always rounded to the nearest whole number).

Important: baseModel and fullModel must be given in that order (simpler model first). If they are given backwards, or the two models are not actually nested, you will get a warning saying so instead of a nonsensical result.

glmerRanova()

Automatic, build-up LRTs for every random effect in a glmer model

Mimics what ranova() does for lmer models—testing each random effect and reporting a likelihood ratio test—but for glmer models, which ranova() does not support. You give it just the one fitted glmer model; it automatically figures out every random effect in it and refits a whole sequence of reduced models (reusing the same data, family, link, and nAGQ as the original model), so no manual nested-model fitting is required. Each row's own "Random Effect Dropped" label is condensed to "Term | Group" form, e.g. "x1 + correlations | schoolID".

For a random-effects term with more than one variance (e.g. a random intercept + slope), the slopes are tested one at a time, in the order they appear in the model's formula, each one compared to a cumulative model built from the intercept and every slope already tested: the first slope's variance plus its correlation with the intercept (2 df), the second slope's variance plus its correlations with the intercept and the first slope (3 df), and so on. As in lmerSummary()'s ranova() table, the intercept of a term that has slopes gets no row of its own. A term with only one variance (e.g. a simple random intercept with no slope) is tested by dropping it entirely, the same way ranova() does.

Example

For a model with a random intercept and two correlated random slopes, formula(1+x1+x2|schoolID), glmerRanova() prints two rows (the intercept of a term with slopes gets no row of its own):

Row	Df	What it tests
`x1 + correlations | schoolID`	df=2	Tests x1's random slope variance, plus its correlation with the intercept.
`x2 + correlations | schoolID`	df=3	Tests x2's random slope variance, plus its correlations with both the intercept and x1.

Basic usage

```
EmptyNon = glmer(data=Example3, family=binomial(link="logit"), nAGQ=7,
                 formula=nonwhite~1+(1|schoolID))
glmerRanova(EmptyNon)
```

Arguments

Argument	Default	What it does
`model`	(required)	The fitted glmer model object, entered without quotes.
`name`	NULL	An optional text label (in quotes) printed above the output.
`LRTdrop`	NULL	Add one extra row testing a custom, user-chosen combination of random effects removed together, all from ONE random-effects dimension -- e.g. LRTdrop=c("x1","x2") to test two correlated random slopes jointly, or LRTdrop="Intercept" for just the random intercept ("(Intercept)", the name lme4 itself uses, also works). Useful because the automatic build-up rows above only ever test one slope (or the next slope's correlations) at a time, so for a 3+-slope correlated block they can't isolate an arbitrary subset jointly. Enter the name(s) in quotes (combined with c() for more than one), matching how each random effect is named in the Model for the Variance table; the row is labeled "Term(s) | Group", the same style as the table's other rows. Its p.mix is left blank: that correction assumes dropping exactly one variance at a time, which doesn't generally hold for an arbitrary combination.
`LRTdropGroup`	NULL	Only needed if the same random-effect name in LRTdrop occurs in more than one random-effects dimension (e.g. x1 as a slope under both schoolID and classroomID) -- name which one you mean (in quotes), matching that dimension's own grouping factor, e.g. LRTdropGroup="schoolID". Leave NULL when there's no such ambiguity. An unmatched name in either argument stops with a clear error rather than guessing.

Argument	Default	What it does
`digits`	3	The number of decimal places shown in the printed table (the LRT statistic and p-values; LL/-2LL/AIC/BIC are always rounded to the nearest whole number).
`leadingBlank`	TRUE	Set to FALSE to suppress this function's own leading blank line -- used internally by glmerSummary(..., LRT=TRUE), which already leaves that blank line before delegating here. Leave as TRUE for a normal, standalone call.

The output table includes both the ordinary p-value and the p.mix mixture-corrected p-value (see the glossary above) for every row, since every row removes at least a variance component.

Note: this only works on glmer models specifically (not lmer, gls, or lme). If you pass it a different model type, it stops with a message pointing you to the right function instead (ranova() or lmerSummary(..., LRT=TRUE) for an lmer model, glsSummary() for a gls model, lmeSummary() for an lme model).

plotResiduals()

Plots of the level-1 residuals, to check normality and constant variance

Draws plots of a model's level-1 (within-cluster) residuals so you can check the two assumptions made about them: that they are normally distributed and that they have constant variance. It is the stand-alone version of the residPlot option in lmerSummary() and lmeSummary(), with extra options for saving the plots to a file and for choosing which kind of residual to plot. It works for lmer and lme models. There are three plots: a histogram of the residuals with a normal curve laid over it; a normal Q-Q plot (points close to the line = approximately normal); and the residuals against the fitted values, with a zero line and a smoothed trend line (an even band around zero = constant variance; a funnel or curve = a problem). The residuals are the model's own conditional level-1 residuals, so the fitted values include the random effects.

If you give by, a fourth plot is added—boxplots of the residuals at each level of that variable (similar box heights = constant variance across the levels)—and the residuals' N, mean, and SD at each level are printed. This is the natural check for repeated-measures data, where the question is whether the variance changes over time.

Basic usage

```
plotResiduals(Model2)                                    # draws in the plot
window
plotResiduals(Model2, by="time_F")                       # adds boxplots of the
residuals at each time
plotResiduals(Model2, by="time_F", file="Residuals.png") # saves a png instead
of drawing it
```

Arguments

Argument	Default	What it does
`model`	(required)	The fitted lmer or lme model object, entered without quotes. (glmer models are not supported, since their residuals are not expected to be normal.)
`by`	NULL	The name, in quotes, of a variable to split the residuals by, e.g. by="time_F". Adds the boxplots by level and the printed N, mean, and SD table. The variable does not have to be in the model, only in the data the model was fit to.
`type`	(automatic)	Which residuals to plot: "response" (raw: observed minus predicted), "pearson" (raw residuals divided by the model's residual SD, including its variance function, so unequal variances modeled with varIdent() are removed), or "normalized" (also removes the estimated within-cluster correlations; lme models only). By default it is "normalized" for an lme model with a residual correlation structure (e.g. corAR1 or corSymm), "pearson" for an lme model with a variance function (e.g. varIdent), and "response" for everything else. The axes say which type was plotted. For a model that fits, normalized residuals look like independent standard normal noise.
`main`	(automatic)	A title to use in place of the default, "Level-1 residuals: " followed by the model's outcome.
`file`	NULL	A png file name, in quotes, to save the plots to instead of drawing them in the plot window.
`width / height / res`	1400 / 1200 / 150	The size (in pixels) and resolution of the png file when file is given.

The function also returns, invisibly, a data frame of each residual, its fitted value, and its by value, in case you want to look at them further.

Note: The plots use the residuals at the lowest level of the model. They do not check the assumptions about the random effects themselves (e.g. that the cluster intercepts are normally distributed).

marginalV()

The model-implied marginal variance-covariance (V) and correlation matrices at times you choose

Builds the model-implied marginal variance-covariance matrix (V) of one cluster's repeated outcomes, and the corresponding marginal correlation matrix, at time values you choose, from a fitted lmer or lme model. V = Z G Z' + R, where G is the random effects variance-covariance matrix, Z holds the random effects' predictors at the chosen times (a column of 1s for the intercept, and the time values for a random time slope), and R is the residual covariance matrix: the residual variance with no correlation for an lmer model, and, for an lme model, also its residual correlation structure (corAR1, corCAR1, corCompSymm, corSymm, or a spatial structure) and any variance function (varIdent, varPower, and so on), evaluated at the chosen times.

The times do not have to be times that anyone was actually observed (unlike nlme's getVarCov(type="marginal"), which is for an individual who was), so you can see how the model implies that the variances and correlations change across, or between, any times you like. Models with more than one clustering variable (for example, occasions nested in persons nested in schools) are not supported, and neither are glmer models.

Basic usage

From Example 5b: random linear change over time, with a continuous-time AR1 (corCAR1) residual correlation.

```
Model3c = lme(data=Example5b, method="REML", fixed=dep~1+time,
              random=~1+time|PersonID,
              correlation=corCAR1(form=~time|PersonID))
marginalV(Model3c, times=c(0, 1, 2, 3))
marginalV(Model3c, times=c(0, 0.5, 3), digits=2)
```

Output of the first call (the second prints the same two matrices for those three times):

```
Model-Implied Marginal Variance-Covariance Matrix (V) for Model3c:
       0      1      2      3
0 11.229  6.340  5.372  4.811
1  6.340 10.834  6.578  6.242
2  5.372  6.578 11.705  8.081
3  4.811  6.242  8.081 13.840

Model-Implied Marginal Correlation Matrix:
      0     1     2     3
0 1.000 0.575 0.469 0.386
1 0.575 1.000 0.584 0.510
2 0.469 0.584 1.000 0.635
3 0.386 0.510 0.635 1.000

(Rows and columns are values of time)
```

Arguments

Argument	Default	What it does
`model`	(required)	The fitted lmer or lme model object, entered without quotes. (glmer models, and models with more than one clustering variable, are not supported.)
`times`	(required)	The time values to compute V at: either a vector, e.g. c(0, 1, 2, 3), or a data frame with one row per occasion and one column for each variable the covariance structure uses. The data frame is needed when, for example, the random slope is on a centered version of time but the residual correlation structure uses the original time, or when a variance function depends on a grouping variable such as occasion.
`timeVar`	NULL	The name, in quotes, of the variable that the values in times are values of, e.g. timeVar="time". If left NULL, it is found from the model (the variable of the random slope and/or the residual

Argument	Default	What it does
		correlation structure) when the model uses just one. If it uses more than one (for example a random slope on a mood variable and a correlation structure on day), give timeVar for the one that times holds; the other is held at 0 (or at its at value) unless it is the variable of the residual correlation structure, in which case give times as a data frame with a column for each.
`at`	NULL	An optional named list of values for any other variable the covariance structure uses, e.g. at=list(sexMW=1) for a model with a varIdent() residual variance by sex. A variable not given in times or at is held at 0 (a factor at its first level, or a varIdent() grouping variable at its reference group), and the values used are listed under the matrices.
`name`	NULL	An optional text label (in quotes) printed above the output.
`digits`	3	The number of decimal places shown in the printed matrices.

The matrices are also returned, invisibly, as V and Corr in a list that includes the pieces V is built from: G (the random effects variance-covariance matrix), R (the residual covariance matrix), Z (the random effects' predictors at the chosen times), and times (the occasions used). For example, x = marginalV(Model3c, times=0:3) lets you use x$V or x$Corr.

Note: corAR1 needs whole-number times (for unequally spaced or non-integer times in an lme model, use corCAR1), and an unstructured residual correlation (corSymm) can only be given at times that were among the occasions the model estimated, since it has a separate correlation for each pair of those occasions.