Package 'lmmresample'

November 19, 2026

Title Resampling Inference for Mixed Models of Densely Sampled Time Series

Version 0.1.0

Description Permutation and bootstrap inference for linear and linear mixed-effects models fitted to block-structured repeated measurements, such as densely sampled physiological time series (pupillometry, eye movements, EEG, fNIRS, heart rate, electrodermal activity, electromyography) or features extracted from them per trial or per participant. Resampling operates on whole trials or participants, so dependence within blocks is preserved. Provides permutation tests under user-declared exchangeability, simulation-based calibration of the permutation null, family-wise error control with the max-t procedure, pairwise comparisons between factor levels, bootstrap confidence intervals (expanded percentile for few clusters, percentile, bias-corrected and accelerated, and basic) and leave-one-subject-out diagnostics.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

Depends R (>= 4.1.0)

Imports cli, future.apply, generics, lme4, Matrix, reformulas, rlang (>= 1.0.0), stats

Suggests future, ggplot2, knitr, progressr, rmarkdown, splines, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/ricardo-martinez-flores/lmmresample

BugReports https://github.com/ricardo-martinez-flores/lmmresample/issues

NeedsCompilation no

Author Ricardo Martínez-Flores [aut, cre] (ORCID: <https://orcid.org/0000-0002-8435-2710>), Hans Supèr [aut]

Maintainer Ricardo Martínez-Flores <ricardo.antonio.martinezf@gmail.com>

Repository CRAN

Date/Publication 2026-11-19 11:50:02 UTC

Contents

boot_ci . . . 3
confint.lmmr_boot . . . 5
diag_acf . . . 6
diag_agreement . . . 7
diag_loso . . . 8
diag_timecourse . . . 9
exchange . . . 10
perm_calibrate . . . 11
perm_maxt . . . 14
perm_pairwise . . . 16
perm_spec . . . 18
perm_test . . . 19
plot.lmmr_acf . . . 22
plot.lmmr_agreement . . . 22
plot.lmmr_boot . . . 23
plot.lmmr_calib . . . 23
plot.lmmr_loso . . . 24
plot.lmmr_maxt . . . 24
plot.lmmr_perm . . . 25
plot.lmmr_timecourse . . . 25
sim_blocks . . . 26
tidy.lmmr_acf . . . 28
tidy.lmmr_agreement . . . 29
tidy.lmmr_boot . . . 30
tidy.lmmr_calib . . . 30
tidy.lmmr_loso . . . 31
tidy.lmmr_maxt . . . 31
tidy.lmmr_pairwise . . . 32
tidy.lmmr_perm . . . 32
tidy.lmmr_timecourse . . . 33

Index **34**

boot_ci *Bootstrap confidence intervals for fixed effects*

Description

Computes bootstrap confidence intervals for the fixed-effect coefficients of a linear model by resampling whole clusters (typically participants), so that any dependence within clusters, such as autocorrelated samples within trials, is preserved.

Usage

```
boot_ci(
  model,
  cluster,
  resample = c("case", "wild"),
  ci = c("expanded", "percentile", "bca", "basic"),
  level = 0.95,
  strata = NULL,
  terms = NULL,
  B = 4999,
  seed = NULL,
  data = NULL
)
```

Arguments

model	A linear model fitted with stats::lm() or lme4::lmer().
cluster	Name of the column identifying the clusters to resample.
resample	Resampling scheme: "case" or "wild". See Details.
ci	Interval type: "expanded" (default), "percentile", "bca" or "basic". See Details.
level	Confidence level.
strata	Optional name of a column that is constant within each cluster (e.g. "group"). Case resampling is then done within its levels. See Details.
terms	Names of the fixed-effect coefficients. By default all except the intercept and the coefficients of terms involving cluster (or grouping factors nested in it), which are not estimable in resampled data.
B	Number of bootstrap replicates. The default, 4999, is suitable for reporting, in particular with BCa intervals.
seed	Optional integer seed. The global random number generator state is left unchanged.
data	The data used to fit model, if it cannot be recovered from the model call.

Details

Two resampling schemes are available:

- `resample = "case"`: clusters are drawn with replacement and the model is refitted to the resampled data. Clusters drawn more than once are given distinct identifiers, as are grouping factors nested within them (e.g. trials), so that each copy is treated as a different participant.
- `resample = "wild"`: the model is refitted to $y^* = X\hat{\beta} + v_c(y - X\hat{\beta})$, where $X\hat{\beta}$ are the fitted values of the fixed effects and $v_c = \pm 1$ is drawn independently for each cluster (Rademacher weights). It keeps the design fixed, is robust to heteroscedasticity and behaves better than the case bootstrap with few clusters.

Four interval types are available, and all replicates are stored, so `confint()` can return another type without refitting (BCa only if the jackknife was computed):

- `"expanded"` (default): the percentile interval taken at a wider nominal level, so that it matches a *t* interval with $G - 1$ degrees of freedom, where G is the number of clusters (expanded percentile interval; Hesterberg, 2015). Plain percentile intervals are too narrow with few clusters; the correction removes most of that shortfall and has little effect with many clusters (about 2% wider with 30 clusters). The limits are the quantiles at $\Phi(\mp\sqrt{G/(G-1)}\, t_{1-\alpha/2,\, G-1})$.
- `"percentile"`: quantiles of the bootstrap distribution.
- `"bca"`: bias-corrected and accelerated (Efron, 1987), with acceleration estimated by a leave-one-cluster-out jackknife. It corrects for bias and skewness and is recommended with at least 2000 replicates. The jackknife requires one extra refit per cluster and is only computed when `ci = "bca"`.
- `"basic"`: reflection of the percentile interval around the estimate.

With `strata`, case resampling draws clusters separately within each stratum, so every bootstrap sample keeps the observed number of clusters per stratum (e.g. 20 controls and 10 patients). Use it when clusters belong to groups that are compared, such as patients and controls: unstratified resampling would produce samples with unbalanced or empty groups. The expanded percentile interval then uses $G - S$ degrees of freedom, where S is the number of strata. The wild bootstrap keeps the design fixed, so strata are not needed there and `strata` is ignored.

The case bootstrap requires clusters to be the top level of the design. When the model contains random effects crossed with `cluster` (e.g. items shared by all participants), the intervals reflect sampling of clusters only and a warning is given; `lme4::bootMer()` provides a parametric alternative.

Value

An object of class `lmmr_boot` with methods for `print()`, `confint()`, tidy() and `plot()`.

References

Hesterberg, T. C. (2015). What teachers should know about the bootstrap: Resampling in the undergraduate statistics curriculum. *The American Statistician, 69*(4), 371-386. doi:10.1080/00031305.2015.1089789

Efron, B. (1987). Better bootstrap confidence intervals. *Journal of the American Statistical Association, 82*(397), 171-185. doi:10.1080/01621459.1987.10478410

Cameron, A. C., Gelbach, J. B., & Miller, D. L. (2008). Bootstrap-based improvements for inference with clustered errors. *The Review of Economics and Statistics, 90*(3), 414-427. doi:10.1162/rest.90.3.414

See Also

`diag_loso()`, `perm_test()`

Examples

```
d <- sim_blocks("within", level = "trial", n_participants = 12,
                n_trials = 10, effect_condition = 0.5, seed = 1)
m <- lm(y ~ condition + participant, data = d)

# B is small here to keep the example fast; use the default for reporting
bt <- boot_ci(m, cluster = "participant", terms = "conditionB", ci = "bca",
              B = 99, seed = 1)
bt
confint(bt, type = "bca")
```

`confint.lmmr_boot` *Confidence intervals from a bootstrap*

Description

Confidence intervals from a bootstrap

Usage

```
## S3 method for class 'lmmr_boot'
confint(object, parm = NULL, level = object$level, type = object$ci, ...)
```

Arguments

`object`	An object of class `lmmr_boot`.
`parm`	Coefficients to return; all by default.
`level`	Confidence level.
`type`	Interval type: `"expanded"`, `"percentile"`, `"bca"` or `"basic"`. Defaults to the type used in `boot_ci()`.
`...`	Not used.

Value

A matrix with one row per coefficient and the lower and upper limits, as returned by `stats::confint()`.

`diag_acf` *Autocorrelation of residuals within series*

Description

Estimates the autocorrelation of the model residuals within each series (e.g. the samples of a trial), ordered by time, and averages it across series. Strong autocorrelation is the reason why Wald inference on densely sampled data is unreliable and why whole series should be resampled. The lag-1 estimate can be passed to `perm_calibrate()` to simulate data with the same dependence.

Usage

```
diag_acf(model, series, time, lag_max = NULL, correct = TRUE, data = NULL)
```

Arguments

`model`	A linear model fitted with `stats::lm()` or `lme4::lmer()`.
`series`	Name of the column identifying each series (e.g. `"trial"`).
`time`	Name of the column giving the order of samples within series.
`lag_max`	Maximum lag. Defaults to 20 or the length of the shortest series minus 2, whichever is smaller.
`correct`	Logical. Return a bias-corrected AR(1) coefficient that accounts for centring short series (see Details)?
`data`	The data used to fit `model`, if it cannot be recovered from the model call.

Details

Each series is centred on its own mean before computing its autocorrelation. This removes series-level offsets (e.g. trial random intercepts) but biases the estimates towards zero, strongly so for short and highly autocorrelated series (for example, a true lag-1 value of 0.92 is estimated as about 0.64 with 20 samples per series). With `correct = TRUE` (the default) `phi` is a bias-corrected AR(1) coefficient: the expected mean autocorrelations of centred AR(1) series with the observed lengths are computed by simulation over a grid of coefficients, and the coefficient whose expected autocorrelations at lags 1 to `lag_max` best match the observed ones (least squares) is returned. Matching several lags, rather than lag 1 only, keeps the estimate sensible when the noise is smooth but not exactly AR(1), as is typical of physiological signals. The simulation uses a fixed internal seed and does not change the random number stream of the session. The autocorrelations in `acf` are reported uncorrected. Series shorter than `lag_max + 2` samples are skipped.

Value

An object of class `lmmr_acf` with methods for `print()`, `tidy()` and `plot()`. Its element `phi` is the AR(1) coefficient (bias-corrected if `correct = TRUE`), `phi_raw` the uncorrected mean lag-1 autocorrelation across series, and `acf` a data frame with the mean, 10th and 90th percentiles of the autocorrelation across series at each lag.

See Also

perm_calibrate(), diag_timecourse()

Examples

```
d <- sim_blocks("within", n_participants = 6, n_trials = 10, n_time = 40,
                ar1 = 0.8, seed = 1)
m <- lm(y ~ condition + participant, data = d)
ac <- diag_acf(m, series = "trial", time = "time")
ac
```

---

diag_agreement *Agreement between the model coefficient and participant-level differences*

Description

Compares the model coefficient of a two-level within-cluster factor (e.g. condition) with the mean of the differences computed directly in each cluster (e.g. the mean of condition B minus the mean of condition A for each participant). Both summarise the same effect, so a large discrepancy suggests that the model is estimating something different from the intended contrast.

Usage

```
diag_agreement(model, term, cluster, tolerance = 0.25, data = NULL)
```

Arguments

model	A linear model fitted with stats::lm() or lme4::lmer().
term	Name of a two-level factor that varies within clusters.
cluster	Name of the column identifying the clusters.
tolerance	Relative discrepancy above which a warning is given.
data	The data used to fit model, if it cannot be recovered from the model call.

Details

A common cause in time-series models is a time course that differs between conditions without a fixed effect of time or a condition by time interaction in the model, so that the condition coefficient absorbs part of the time course. Discrepancies can also arise legitimately: covariates in the model, unequal numbers of trials per participant (the model weights participants by their precision) or strong shrinkage of random slopes. The check flags a discrepancy only when it exceeds both tolerance (relative to the mean difference) and the standard error of the mean difference.

Value

An object of class lmmr_agreement with methods for print(), tidy() and plot().

See Also

diag_timecourse()

Examples

```
d <- sim_blocks("within", n_participants = 10, n_trials = 10, n_time = 20,
                effect_condition = 0.5, seed = 1)
m <- lme4::lmer(y ~ condition + time + (1 | participant), data = d)
diag_agreement(m, term = "condition", cluster = "participant")
```

---

diag_loso *Leave-one-cluster-out influence*

Description

Refits the model leaving out each cluster (e.g. participant) in turn and reports how much each fixed-effect estimate changes. Large changes flag influential participants whose removal would alter the conclusions.

Usage

```
diag_loso(model, cluster, terms = NULL, data = NULL)
```

Arguments

model	A linear model fitted with stats::lm() or lme4::lmer().
cluster	Name of the column identifying the clusters to leave out.
terms	Names of the fixed-effect coefficients to report. By default all except the intercept and the coefficients of terms involving cluster.
data	The data used to fit model, if it cannot be recovered from the model call.

Details

The change is also expressed in units of the standard error of the full model estimate. As a rough guide, values above 1 deserve inspection. Refits run in parallel through the **future** framework.

Value

An object of class lmmr_loso with methods for print(), tidy() and plot(). Its element estimates is a matrix with one row per left-out cluster and one column per coefficient.

See Also

boot_ci()

Examples

```
d <- sim_blocks("within", level = "trial", n_participants = 10,
                n_trials = 10, effect_condition = 0.5, seed = 1)
m <- lm(y ~ condition + participant, data = d)
loso <- diag_loso(m, cluster = "participant", terms = "conditionB")
loso
```

`diag_timecourse` *Residual time courses by condition*

Description

Averages the model residuals over time, separately for each level of a grouping variable such as the condition. If the model describes the time course well, the mean residual stays close to zero at all times in every condition. Systematic departures indicate that the shape of the response (e.g. a linear term for a curved response) or its difference between conditions (a missing condition by time interaction) is misspecified.

Usage

```
diag_timecourse(model, time, by = NULL, cluster = NULL, bins = 30, data = NULL)
```

Arguments

`model`	A linear model fitted with `stats::lm()` or `lme4::lmer()`.
`time`	Name of the column giving the order of samples within series.
`by`	Optional name of a grouping column (e.g. `"condition"`).
`cluster`	Optional name of the column identifying independent clusters (e.g. `"participant"`), used for the standard errors.
`bins`	Number of time bins used when `time` has more than 50 distinct values. `NULL` keeps the original time points.
`data`	The data used to fit `model`, if it cannot be recovered from the model call.

Details

A misspecified time course does not invalidate permutation tests of the model terms, but the coefficients may then not answer the intended question and the tests lose power. Flexible terms such as `splines::ns(time, df)` and their interaction with condition are usually adequate. The mean residual and its standard error are computed across clusters (e.g. participants) when `cluster` is given, and across rows otherwise. A departure is flagged when the largest |mean / SE| exceeds the Bonferroni-corrected two-sided 5% threshold over all time points and groups; this assumes clusters are independent, so give `cluster`.

Value

An object of class `lmmr_timecourse` with methods for `print()`, tidy() and `plot()`.

See Also

diag_acf()

Examples

```
d <- sim_blocks("within", n_participants = 8, n_trials = 10, n_time = 30,
                effect_condition = 1, effect_onset = 0.3, seed = 1)
m <- lm(y ~ condition + time + participant, data = d)
tc <- diag_timecourse(m, time = "time", by = "condition",
                      cluster = "participant")
tc
```

---

exchange *Declare exchangeability for permutation tests*

Description

These functions describe which units may be exchanged when building the permutation distribution. They are passed to the `exchange` argument of `perm_test()`, together with `unit`, the column identifying the units whose labels are permuted.

Usage

```
exch_within(block)

exch_signflip(block)

exch_free(strata = NULL)
```

Arguments

`block`	Name of the column defining the blocks within which units are exchanged (or whose labels are swapped), typically the participant identifier.
`strata`	Optional name of a column defining strata within which units are exchanged (e.g. recording site). `NULL` (the default) exchanges units freely.

Details

- `exch_within()`: units are exchanged only within each level of `block`. Use it for within-participant designs, where trials (`unit`) are permuted within each participant (`block`).
- `exch_free()`: units are exchanged freely across the whole data set, or within each level of `strata` if supplied. Use it for between-group designs, where participants (`unit`) are permuted across groups.

- `exch_signflip()`: the two levels of the tested variable are swapped for all units of a randomly chosen subset of blocks, which reverses the sign of each selected participant's effect. Use it in within-participant designs when participants may differ in the size of their effect (random slopes) and the hypothesis concerns the mean effect. It assumes that individual effects are symmetrically distributed around the mean under the null hypothesis.

`exch_within()` is exact when the tested variable has no effect in any participant (sharp null). When participants differ in their effect, it can be liberal for the mean effect; `exch_signflip()` is then preferable. Use `perm_calibrate()` to check either choice for a given design.

Each unit must belong to a single block (or stratum), and the permuted variable must be constant within each unit.

Value

An object of class `lmmr_exchange`.

Examples

```
# Within-participant design: permute trials within participants
exch_within("participant")

# Between-group design: permute participants freely
exch_free()

# Between-group design stratified by site
exch_free(strata = "site")

# Within-participant design, inference on the mean effect when
# participants differ in their effect
exch_signflip("participant")
```

---

perm_calibrate *Calibrate a permutation test by simulation*

Description

Checks whether `perm_test()` controls the type I error rate for the user's own design and model. Data are simulated repeatedly under the null hypothesis from the fitted model, with the tested coefficient set to zero, and the permutation test is applied to each simulated data set. If the test is well calibrated, the proportion of rejections is close to `alpha` and the p-values are uniformly distributed. For comparison, the rejection rate of the model's Wald test is reported for the same simulated data sets.

Usage

```
perm_calibrate(
  model,
  term,
  unit,
```

```
  exchange,
  coef = NULL,
  method = c("auto", "relabel", "freedman-lane"),
  null = c("mean", "sharp"),
  ar1 = NULL,
  series = NULL,
  time = NULL,
  n_sim = 200,
  B = 199,
  alpha = 0.05,
  seed = NULL,
  data = NULL
)
```

Arguments

`model`	A linear model fitted with `stats::lm()` or `lme4::lmer()`.
`term`	Fixed-effect term to test, as labelled in the model formula (e.g. `"condition"` or `"condition:time"`). For `method = "relabel"` it must be a column of the data, constant within each `unit`.
`unit`	Name of the column identifying the units whose values of `term` are permuted (e.g. `"trial"` or `"participant"`).
`exchange`	Exchangeability of units, created with `exch_within()`, `exch_signflip()` or `exch_free()`.
`coef`	Optional name of the coefficient(s) of `term` to test. By default all coefficients of the term are tested jointly.
`method`	Resampling method, as in `perm_test()`.
`null`	Null hypothesis under which data are simulated: `"mean"` or `"sharp"`. See Details.
`ar1`	Optional lag-1 autocorrelation of the simulated residuals within each `series`, in $[0, 1)$.
`series, time`	Names of the columns identifying each series (e.g. `"trial"`) and the time order within it. Required when `ar1` is supplied; `time` is also used to match samples when Freedman-Lane residuals are permuted between units (see `perm_test()`).
`n_sim`	Number of simulated data sets.
`B`	Number of permutations per simulated data set.
`alpha`	Significance level at which rejection rates are evaluated.
`seed`	Optional integer seed for reproducible permutations. The global random number generator state is left unchanged.
`data`	The data used to fit `model`. Needed only if it cannot be recovered from the model call.

Details

Simulated responses keep everything else from the fitted model: the remaining fixed effects, the estimated random-effects covariance and the residual variance. Two null hypotheses are available:

- `null = "mean"` (default): the average effect of `term` is zero, but participants differ in their individual effect as estimated by the random slope of `term`. This is the hypothesis usually claimed in applied work, and the one under which a permutation test can fail, so it is the informative check.
- `null = "sharp"`: the effect of `term` is zero for every participant; the random slope of `term` is also set to zero. The permutation test is exact under this null when units are exchangeable, so it serves as a reference.

Residuals are simulated as independent by default. For densely sampled series, `ar1` adds first-order autocorrelation within each `series` (e.g. trial), ordered by `time`, keeping the residual standard deviation of the model. This reproduces the dependence that makes standard inference unreliable for such data.

The check is only as realistic as the fitted model: anything the model leaves in the residuals is simulated as random noise. In particular, a common time course that the model does not capture (e.g. the average response curve, fitted only as a linear trend) is treated as noise that varies from trial to trial, which can make the simulated data behave quite differently from the real data. Fit the time course flexibly (e.g. with splines or time as a factor) before calibrating; residual time courses can be inspected with `diag_timecourse()`.

Each simulation runs a complete permutation test, so the total number of model fits is `n_sim * (B + 1)`. Simulations run in parallel through the **future** framework (see `perm_test()`). The defaults give a quick check; for a definitive calibration increase `n_sim` (e.g. 1000), which narrows the confidence interval of the rejection rate.

Value

An object of class `lmmr_calib` with methods for `print()`, tidy() and `plot()`. Its elements include `p_perm` and `p_wald` (p-values of the permutation and Wald tests in each simulation) and `rejection` (rejection rates at `alpha` with exact binomial confidence intervals).

See Also

`perm_test()`

Examples

```
d <- sim_blocks("within", level = "trial", n_participants = 10,
                n_trials = 8, seed = 1)
m <- lm(y ~ condition + participant, data = d)

# Small values to keep the example fast
cal <- perm_calibrate(m, term = "condition", unit = "trial",
                      exchange = exch_within("participant"),
                      n_sim = 20, B = 49, seed = 1)
cal
```

perm_maxt *Family-wise permutation tests (max-t and min-p)*

Description

Tests several model terms at once, for example the same condition effect in one model per feature extracted from the same trials, and controls the family-wise error rate with the max-t or min-p procedures of Westfall and Young (1993). In each permutation the same resampling of units is applied to every model, so the dependence among the test statistics is preserved and the correction is less conservative than Bonferroni when the statistics are correlated.

Usage

```
perm_maxt(
  ...,
  unit,
  exchange,
  method = c("auto", "relabel", "freedman-lane"),
  time = NULL,
  combine = c("auto", "max-t", "min-p"),
  adjust = c("step-down", "single-step"),
  B = 4999,
  seed = NULL
)
```

Arguments

`...`	Named test specifications created with `perm_spec()`. Names are used to label the tests.
`unit`, `exchange`, `time`, `B`, `seed`	As in `perm_test()`.
`method`	Resampling method: `"auto"`, `"relabel"` or `"freedman-lane"`. See Details.
`combine`	`"auto"`, `"max-t"` or `"min-p"`. See Details.
`adjust`	`"step-down"` (default) or `"single-step"`. See Details.

Details

All models must be fitted to data that share the resampled units (e.g. the same trials of the same participants). Two resampling methods are available, as in `perm_test()`:

- `"relabel"`: the values of the tested variable are reassigned among units. All specifications must then test the same variable, with the same values in every data set.
- `"freedman-lane"`: the residuals of each reduced model are permuted or sign-flipped with the same draw for every model. Specifications may test different terms, including interactions and effects adjusted for covariates.

With `method = "auto"`, relabelling is used when it is valid for every specification, and Freedman-Lane otherwise.

Two ways of combining the tests are available:

- `combine = "max-t"`: in each permutation the largest statistic across tests is recorded. The statistics must be on the same scale, so every term must have a single coefficient (absolute *t* statistics).
- `combine = "min-p"`: each statistic is first converted into a p-value within its own permutation distribution, and the smallest p-value across tests is recorded. It accepts statistics on different scales, such as a *t* for a main effect and a chi-square for a condition by spline-of-time interaction, and gives each test the same weight.

With `combine = "auto"`, max-t is used when every term has a single coefficient and min-p otherwise.

- `adjust = "single-step"`: the adjusted p-value of each test is the proportion of permutations whose maximum statistic (or minimum p-value) is at least as extreme as the observed one for that test.
- `adjust = "step-down"`: tests are ordered from the most to the least extreme and the maximum (or minimum) is taken only over the tests not yet rejected, with monotonicity enforced. It is uniformly more powerful than the single-step procedure and controls the family-wise error rate under the same conditions.

Adjusted p-values use the Phipson and Smyth (2010) correction. Permutations in which any model failed to fit are excluded and counted. Tests are two-sided.

Value

An object of class `lmmr_maxt` with methods for `print()`, tidy() and `plot()`. Its element `results` is a data frame with the observed statistic, the unadjusted permutation p-value and the family-wise adjusted p-value of each test; `null` is the matrix of permuted statistics (one column per test).

References

Westfall, P. H., & Young, S. S. (1993). *Resampling-based multiple testing: Examples and methods for p-value adjustment*. Wiley.

Phipson, B., & Smyth, G. K. (2010). Permutation p-values should never be zero. *Statistical Applications in Genetics and Molecular Biology, 9*(1), Article 39. doi:10.2202/15446115.1585

See Also

`perm_spec()`, `perm_test()`

Examples

```
d <- sim_blocks("within", level = "trial", n_participants = 10,
                n_trials = 8, effect_condition = 0.5, seed = 1)
m_mean <- lm(y ~ condition + participant, data = d)
m_peak <- lm(peak ~ condition + participant, data = d)
```

```
# B is small here to keep the example fast; use the default for reporting
res <- perm_maxt(
  mean = perm_spec(m_mean, "condition"),
  peak = perm_spec(m_peak, "condition"),
  unit = "trial", exchange = exch_signflip("participant"),
  B = 99, seed = 1
)
res
```

`perm_pairwise` *Pairwise comparisons between the levels of a factor*

Description

Compares pairs of levels of a factor (post hoc tests) with Freedman-Lane permutation tests, controls the family-wise error rate across pairs, and optionally adds bootstrap confidence intervals of each pairwise difference.

Usage

```
perm_pairwise(
  model,
  factor,
  term = factor,
  unit,
  exchange,
  pairs = NULL,
  adjust = c("westfall-young", "holm", "none"),
  time = NULL,
  B = 4999,
  seed = NULL,
  cluster = NULL,
  strata = NULL,
  ci = c("expanded", "percentile", "bca", "basic"),
  level = 0.95,
  data = NULL
)
```

Arguments

`model`	A model fitted with `stats::lm()` or `lme4::lmer()` that contains `factor`.
`factor`	Name of the factor (or character column) whose levels are compared.
`term`	The model term compared between levels: `factor` (default) or an interaction containing it.
`unit`, `exchange`, `time`, `B`, `seed`	As in `perm_test()`.

`pairs` Pairs of levels to compare: a two-column character matrix (or a list of length-two character vectors). Each row gives the reference level and the compared level. Default: all pairs, in the order of the factor levels.

`adjust` Family-wise adjustment: `"westfall-young"` (default; max-t or min-p, step-down), `"holm"` or `"none"`.

`cluster` Optional cluster column for bootstrap confidence intervals (e.g. `"participant"`). `NULL` (default) skips the intervals.

`strata`, `ci`, `level` Passed to `boot_ci()`.

`data` The data used to fit `model`, if it cannot be recovered from the model call.

Details

For each pair, the model is refitted with the factor coded by indicator (dummy) variables whose reference is the first level of the pair; this reparametrisation leaves the fit unchanged. The pair is tested through the indicator of its second level, while the indicators of the other levels remain in the reduced model of the Freedman-Lane procedure. A difference involving other levels therefore does not leak into the test of the pair, as it would if all labels of the factor were permuted together.

`term` sets what is compared: the factor itself (difference in level) or an interaction that contains it, such as `"condition:ns(time, 3)"` (difference in time course). Random slopes of the factor are reparametrised in the same way.

All pairs are tested on the same data with the same permutations, so they form one family, adjusted by default with the step-down procedure of Westfall and Young (1993): max-t when each pair is a single coefficient, min-p otherwise (see `perm_maxt()`). Holm's method on the unadjusted permutation p-values is also available.

Confidence intervals (when `cluster` is given) are bootstrap intervals of each pairwise difference from `boot_ci()`, with case resampling of `cluster`; they are available when each pair is a single coefficient. They are not adjusted for multiplicity.

Value

An object of class `lmmr_pairwise` with methods for `print()` and tidy(). Its element `results` is a data frame with one row per pair: the levels, the estimated difference (compared minus reference level; single-coefficient comparisons only), the statistic, the unadjusted and adjusted permutation p-values and, if requested, the confidence limits. The family-wise test is stored in `maxt`, and the reparametrised models (one per reference level) in `models`.

References

Westfall, P. H., & Young, S. S. (1993). *Resampling-based multiple testing: Examples and methods for p-value adjustment*. Wiley.

See Also

`perm_test()` for the overall test of the factor, `perm_maxt()`, `boot_ci()`

Examples

```
set.seed(1)
d <- sim_blocks("within", level = "trial", n_participants = 8,
                n_trials = 12, seed = 1)
d$stimulus <- factor(sample(c("A", "B", "C"), nrow(d), replace = TRUE))
d$y <- d$y + 0.8 * (d$stimulus == "C")
m <- lm(y ~ stimulus + participant, data = d)

# B is small here to keep the example fast; use the default for reporting
perm_pairwise(m, "stimulus", unit = "trial",
              exchange = exch_signflip("participant"), B = 99, seed = 1)
```

`perm_spec` *Specify one test for a family of permutation tests*

Description

Pairs a fitted model with the term tested in it, for use in `perm_maxt()`.

Usage

```
perm_spec(model, term, coef = NULL, data = NULL)
```

Arguments

`model`	A linear model fitted with `stats::lm()` or `lme4::lmer()`.
`term`	Fixed-effect term to test, as labelled in the model formula (e.g. `"condition"` or `"condition:time"`). For `method = "relabel"` it must be a column of the data, constant within each `unit`.
`coef`	Optional name of the coefficient(s) of `term` to test. By default all coefficients of the term are tested jointly.
`data`	The data used to fit `model`. Needed only if it cannot be recovered from the model call.

Value

An object of class `lmmr_spec`.

See Also

`perm_maxt()`

Examples

```
d <- sim_blocks("within", level = "trial", n_participants = 6,
                n_trials = 6, seed = 1)
perm_spec(lm(y ~ condition + participant, data = d), "condition")
```

perm_test *Permutation test for a model term*

Description

Tests a fixed-effect term by resampling whole units (trials or participants) and refitting the full model to each resampled data set. Because complete units are exchanged, any dependence within a unit, such as the autocorrelation of samples within a trial, is preserved in the permutation distribution without having to be modelled.

Usage

```
perm_test(
  model,
  term,
  unit,
  exchange,
  coef = NULL,
  method = c("auto", "relabel", "freedman-lane"),
  time = NULL,
  alternative = c("two.sided", "greater", "less"),
  B = 4999,
  seed = NULL,
  data = NULL
)
```

Arguments

model	A linear model fitted with stats::lm() or lme4::lmer().
term	Fixed-effect term to test, as labelled in the model formula (e.g. "condition" or "condition:time"). For method = "relabel" it must be a column of the data, constant within each unit.
unit	Name of the column identifying the units whose values of term are permuted (e.g. "trial" or "participant").
exchange	Exchangeability of units, created with exch_within(), exch_signflip() or exch_free().
coef	Optional name of the coefficient(s) of term to test. By default all coefficients of the term are tested jointly.
method	Resampling method: "auto", "relabel" or "freedman-lane". See Details.
time	Name of the column giving the order of samples within units. Required when Freedman-Lane residuals are permuted between units of several rows (with exch_within() or exch_free()), so that samples are matched by time; not needed with exch_signflip().
alternative	Direction of the alternative hypothesis.
B	Number of permutations. The default, 4999, is suitable for reporting; use smaller values only for exploration.

`seed`	Optional integer seed for reproducible permutations. The global random number generator state is left unchanged.
`data`	The data used to fit `model`. Needed only if it cannot be recovered from the model call.

Details

Two resampling methods are available:

- `"relabel"`: the values of `term` are reassigned among units, as declared by `exchange`. It requires `term` to be a variable constant within units and is used for main effects.
- `"freedman-lane"`: the reduced model without `term` is fitted, and its residuals are permuted between whole units (or sign-flipped by block with `exch_signflip()`) and added back to its fitted values (Freedman & Lane, 1983; Winkler et al., 2014). It accounts for nuisance terms and allows tests of interactions (e.g. `"condition:time"`) and of terms with several coefficients. Random slopes of `term` are removed from the reduced model, because they would absorb the tested effect. So are random slopes of a lower-order term contained in `term` when the other variables of `term` are constant within the grouping factor: when testing a between-participant `group:condition` interaction, the slope in `(1 + condition | participant)` would absorb the group difference in condition effects. Permuting residuals between units requires units of equal size within blocks; sign-flipping accepts units of any size.

With `method = "auto"` (default), relabelling is used for a main effect that is a column of the data with a single coefficient, and Freedman-Lane otherwise.

The statistic is the Wald statistic of the coefficient (its t value: estimate divided by its standard error). When the term has several coefficients (a factor with more than two levels, or an interaction with a spline of time), the statistic is the Wald chi-square of all of them and the test is two-sided. Refits keep every argument of the original call; only the data change.

The p-value follows Phipson and Smyth (2010):

$$p = (b + 1)/(B' + 1),$$

where b is the number of permutations with a statistic at least as extreme as the observed one and B' is the number of permutations that were refitted successfully. Refits that fail or do not converge are excluded and counted; singular fits are kept and counted.

For mixed models, each refit starts from the variance parameters of the original model and skips the derivative-based convergence check of **lme4**, which roughly halves the time per refit on large data sets. Set `options(lmmresample.fast = FALSE)` to refit with the default settings; a `control` argument in the original model call is always respected.

Permutations are generated in the main R session, so results are identical for a given `seed` whatever the parallel backend. Refits run in parallel through the **future** framework: choose a backend with `future::plan()`, e.g. `future::plan("multisession", workers = 4)`. Progress can be reported with `progressr::with_progress()`.

Terms contained in a higher-order interaction are refused (test the interaction first), unless every variable they interact with is numeric and centred. This is the case in growth curve analysis with orthogonal polynomials of time (e.g. `poly(time, 3)`): the condition term is then the average difference over the analysed window and is tested with Freedman-Lane, while the condition-by-polynomial terms test differences in the shape of the curve.

Value

An object of class `lmmr_perm` with methods for `print()`, `summary()`, tidy() and `plot()`. Its elements include:

`statistic`	Observed statistic.
`null`	Statistics of all `B` permutations (`NA` for excluded refits).
`status`	Refit status of each permutation: `"ok"`, `"singular"`, `"nonconverged"` or `"failed"`.
`p.value`	Permutation p-value.
`B`, `B_used`	Requested and successfully used permutations.

References

Freedman, D., & Lane, D. (1983). A nonstochastic interpretation of reported significance levels. *Journal of Business & Economic Statistics, 1*(4), 292-298. doi:10.1080/07350015.1983.10509354

Winkler, A. M., Ridgway, G. R., Webster, M. A., Smith, S. M., & Nichols, T. E. (2014). Permutation inference for the general linear model. *NeuroImage, 92*, 381-397. doi:10.1016/j.neuroimage.2014.01.060

Phipson, B., & Smyth, G. K. (2010). Permutation p-values should never be zero: Calculating exact p-values when permutations are randomly drawn. *Statistical Applications in Genetics and Molecular Biology, 9*(1), Article 39. doi:10.2202/15446115.1585

See Also

`exch_within()`, `exch_signflip()`, `exch_free()`, `perm_calibrate()`

Examples

```
d <- sim_blocks("within", level = "trial", n_participants = 12,
                n_trials = 10, effect_condition = 0.4, seed = 1)
m <- lm(y ~ condition + participant, data = d)

# B is small here to keep the example fast; use the default for reporting
res <- perm_test(m, term = "condition", unit = "trial",
                 exchange = exch_within("participant"), B = 199, seed = 1)
res


# Mixed model on the full time series
if (requireNamespace("lme4", quietly = TRUE)) {
  ts <- sim_blocks("within", n_participants = 10, n_trials = 10,
                   n_time = 50, effect_condition = 0.4, seed = 2)
  m_ts <- lme4::lmer(y ~ condition + (1 | participant), data = ts)
  perm_test(m_ts, term = "condition", unit = "trial",
            exchange = exch_within("participant"), B = 99, seed = 1)

  # When does the effect appear? Condition x time interaction
  ts2 <- sim_blocks("within", n_participants = 10, n_trials = 10,
                    n_time = 30, effect_condition = 0.5,
                    effect_onset = 0.3, seed = 3)
  m_int <- lme4::lmer(y ~ condition * time + (1 | participant), data = ts2)
```

```
  perm_test(m_int, term = "condition:time", unit = "trial",
            exchange = exch_signflip("participant"), B = 99, seed = 1)
}
```

`plot.lmmr_acf` *Plot a residual autocorrelation diagnostic*

Description

Plot a residual autocorrelation diagnostic

Usage

```
## S3 method for class 'lmmr_acf'
plot(x, ...)
```

Arguments

`x`	An object of class `lmmr_acf`.
`...`	Not used.

Value

A `ggplot2::ggplot()` object.

`plot.lmmr_agreement` *Plot an agreement diagnostic*

Description

Plot an agreement diagnostic

Usage

```
## S3 method for class 'lmmr_agreement'
plot(x, ...)
```

Arguments

`x`	An object of class `lmmr_agreement`.
`...`	Not used.

Value

A `ggplot2::ggplot()` object.

plot.lmmr_boot *Plot a bootstrap*

Description

Shows the bootstrap distribution of each coefficient with the estimate, the expanded percentile and percentile intervals, and the BCa interval if it was computed, so that their differences (due to few clusters, bias or skewness) can be seen.

Usage

```
## S3 method for class 'lmmr_boot'
plot(x, ...)
```

Arguments

x	An object of class lmmr_boot.
...	Not used.

Value

A ggplot2::ggplot() object with one panel per coefficient.

plot.lmmr_calib *Plot a calibration*

Description

Plot a calibration

Usage

```
## S3 method for class 'lmmr_calib'
plot(x, type = c("pp", "rejection"), ...)
```

Arguments

x	An object of class lmmr_calib.
type	"pp" compares the observed p-values with the uniform distribution expected under a well-calibrated test, with a pointwise 95% band. "rejection" shows the rejection rate of each method with its confidence interval.
...	Not used.

Value

A ggplot2::ggplot() object.

plot.lmmr_loso *Plot a leave-one-cluster-out diagnostic*

Description

Plot a leave-one-cluster-out diagnostic

Usage

```
## S3 method for class 'lmmr_loso'
plot(x, ...)
```

Arguments

x	An object of class lmmr_loso.
...	Not used.

Value

A ggplot2::ggplot() object with one panel per coefficient.

plot.lmmr_maxt *Plot family-wise permutation tests*

Description

Shows the permutation distribution of the maximum absolute statistic (max-t) or of the minimum p-value (min-p) across tests, with the observed value of each test and the critical value that controls the family-wise error rate at alpha.

Usage

```
## S3 method for class 'lmmr_maxt'
plot(x, alpha = 0.05, ...)
```

Arguments

x	An object of class lmmr_maxt.
alpha	Family-wise significance level for the critical value.
...	Not used.

Value

A ggplot2::ggplot() object.

plot.lmmr_perm *Plot a permutation test*

Description

Plot a permutation test

Usage

```
## S3 method for class 'lmmr_perm'
plot(x, type = c("null", "trace"), ...)
```

Arguments

x	An object of class lmmr_perm.
type	"null" shows the permutation distribution with the observed statistic. "trace" shows the running p-value as permutations accumulate, to check that B is large enough for a stable result.
...	Not used.

Value

A ggplot2::ggplot() object.

plot.lmmr_timecourse *Plot a residual time-course diagnostic*

Description

Plot a residual time-course diagnostic

Usage

```
## S3 method for class 'lmmr_timecourse'
plot(x, ...)
```

Arguments

x	An object of class lmmr_timecourse.
...	Not used.

Value

A ggplot2::ggplot() object.

`sim_blocks` *Simulate block-structured repeated measurements*

Description

Generates data with a known effect structure in which densely sampled series (e.g. pupil size over time) are recorded in trials nested within participants. The data can be returned at three levels of aggregation: the full time series, one row per trial, or one row per participant (and condition). Used in examples, tests and vignettes, and useful for power analysis and for checking an analysis pipeline before applying it to real data.

Usage

```
sim_blocks(
  design = c("within", "between", "mixed"),
  level = c("timeseries", "trial", "participant"),
  n_participants = 30,
  n_trials = 40,
  n_time = 100,
  sampling_rate = 50,
  effect_condition = 0,
  effect_group = 0,
  effect_interaction = 0,
  sd_participant = 1,
  sd_slope = 0,
  sd_trial = 0.5,
  sd_residual = 1,
  ar1 = 0.9,
  prop_condition = 0.5,
  effect_onset = 0,
  ar1_trials = 0,
  residual_df = Inf,
  seed = NULL
)
```

Arguments

`design`	Design of the study. `"within"`: every participant completes trials in both conditions. `"between"`: participants belong to one of two groups and there is no condition factor. `"mixed"`: both.
`level`	Level of aggregation of the returned data. `"timeseries"`: one row per sample. `"trial"`: one row per trial, with the mean (`y`) and the maximum (`peak`) of each series. `"participant"`: trial-level values averaged within each participant (and condition, if present).
`n_participants`	Number of participants. In `"between"` and `"mixed"` designs they are split as evenly as possible between the two groups.

n_trials	Number of trials per participant: a single number, or a vector with one value per participant to simulate participants with different numbers of trials (e.g. after trial rejection). In "within" and "mixed" designs, a proportion prop_condition of them is assigned to condition "B" in random order; with balanced conditions (the default) n_trials must be even.
n_time	Number of samples per trial.
sampling_rate	Sampling rate in Hz, used to express time in seconds.
effect_condition	Difference between the second and first level of condition ("B" minus "A").
effect_group	Difference between the second and first level of group ("treatment" minus "control").
effect_interaction	Additional effect of condition "B" in group "treatment" (mixed designs only).
sd_participant	Standard deviation of participant random intercepts.
sd_slope	Standard deviation of participant random slopes for condition. Values greater than zero mean that participants differ in their condition effect.
sd_trial	Standard deviation of trial random intercepts.
sd_residual	Marginal standard deviation of the within-trial residuals.
ar1	Lag-1 autocorrelation of the within-trial residuals, in $[0, 1)$.
prop_condition	Proportion of each participant's trials assigned to condition "B" ("within" and "mixed" designs). The default, 0.5, gives balanced conditions; e.g. 0.2 mimics an oddball design with 20% targets. The number of "B" trials is rounded to the nearest integer and kept between 1 and n_trials - 1.
effect_onset	Time (in seconds from trial onset) at which condition effects (effect_condition, effect_interaction and the random slopes) appear. The default, 0, gives effects that are constant over the trial; positive values give a time-varying effect, as in pupil responses that diverge only some hundreds of milliseconds after the stimulus.
ar1_trials	Correlation between the random intercepts of consecutive trials of the same participant, in $[0, 1)$, to simulate dependence between trials (e.g. slow drifts or carry-over).
residual_df	Degrees of freedom of a Student t distribution for the innovations of the within-trial residuals. Inf (the default) gives normal residuals; small values (e.g. 4 or 5) give heavy-tailed residuals. Residuals are rescaled so that their marginal standard deviation is sd_residual.
seed	Optional integer seed. If supplied, the simulation is reproducible and the global random number generator state is left unchanged.

Value

A data frame. Columns depend on design and level:

- participant: participant identifier (factor).
- group: "control" or "treatment" ("between" and "mixed").

- `trial`: unique trial identifier (factor; `"timeseries"` and `"trial"` levels).
- `trial_index`: position of the trial within the participant's session.
- `condition`: `"A"` or `"B"` (`"within"` and `"mixed"`).
- `time`: time from trial onset in seconds (`"timeseries"` level).
- `y`: the signal (`"timeseries"`) or its mean over the trial.
- `peak`: maximum of the series within the trial (`"trial"` and `"participant"` levels).

The simulation parameters are stored in the attribute `"params"`.

Generative model

For participant i, trial j and sample t,

$$y_{ijt} = s(t) + w(t)(\beta_c c_{ij} + \beta_{cg} c_{ij} g_i + b_{1i} c_{ij}) + \beta_g g_i + b_{0i} + u_{ij} + e_{ijt},$$

where $s(t)$ is a smooth response curve common to all trials, c_{ij} and g_i are 0/1 indicators for the second level of `condition` and `group`, $b_{0i} \sim N(0, \sigma_p^2)$ and $b_{1i} \sim N(0, \sigma_s^2)$ are participant random intercepts and condition slopes, u_{ij} is a trial random intercept with standard deviation σ_t (optionally following an AR(1) process across consecutive trials, `ar1_trials`), and e_{ijt} follows a stationary AR(1) process within each trial with coefficient `ar1` and marginal standard deviation `sd_residual`. The weight $w(t)$ is 1 at all times by default; with `effect_onset > 0` it rises smoothly from 0 to 1 around the onset, so condition effects appear only later in the trial.

With the default constant weight, in a model containing `condition` (and/or `group`) the corresponding fixed-effect coefficient estimates `effect_condition` (and/or `effect_group`) directly.

Examples

```
# Full time series, within-participant design
d <- sim_blocks("within", n_participants = 10, n_trials = 20,
                effect_condition = 0.3, seed = 1)
head(d)

# One row per trial
d_trial <- sim_blocks("within", level = "trial", seed = 1)

# One row per participant, between-group design
d_part <- sim_blocks("between", level = "participant",
                     effect_group = 0.5, seed = 1)
aggregate(y ~ group, data = d_part, FUN = mean)
```

---

tidy.lmmr_acf *Tidy a residual autocorrelation diagnostic*

Description

Tidy a residual autocorrelation diagnostic

Usage

```
## S3 method for class 'lmmr_acf'
tidy(x, ...)
```

Arguments

x	An object of class lmmr_acf.
...	Not used.

Value

A data frame with one row per lag.

---

tidy.lmmr_agreement *Tidy an agreement diagnostic*

Description

Tidy an agreement diagnostic

Usage

```
## S3 method for class 'lmmr_agreement'
tidy(x, ...)
```

Arguments

x	An object of class lmmr_agreement.
...	Not used.

Value

A data frame with one row per cluster and its difference.

tidy.lmmr_boot *Tidy a bootstrap*

Description

Tidy a bootstrap

Usage

```
## S3 method for class 'lmmr_boot'
tidy(x, type = c("summary", "replicates"), ...)
```

Arguments

x	An object of class lmmr_boot.
type	"summary" returns the estimates and intervals of the type used in boot_ci(); "replicates" returns all bootstrap estimates in long format.
...	Not used.

Value

A data frame.

tidy.lmmr_calib *Tidy a calibration*

Description

Tidy a calibration

Usage

```
## S3 method for class 'lmmr_calib'
tidy(x, type = c("summary", "pvalues"), ...)
```

Arguments

x	An object of class lmmr_calib.
type	"summary" returns the rejection rates; "pvalues" returns the p-values of each simulation.
...	Not used.

Value

A data frame.

tidy.lmmr_loso *Tidy a leave-one-cluster-out diagnostic*

Description

Tidy a leave-one-cluster-out diagnostic

Usage

```
## S3 method for class 'lmmr_loso'
tidy(x, ...)
```

Arguments

x	An object of class lmmr_loso.
...	Not used.

Value

A data frame with one row per left-out cluster and coefficient.

tidy.lmmr_maxt *Tidy family-wise permutation tests*

Description

Tidy family-wise permutation tests

Usage

```
## S3 method for class 'lmmr_maxt'
tidy(x, type = c("summary", "null"), ...)
```

Arguments

x	An object of class lmmr_maxt.
type	"summary" returns one row per test; "null" returns the permuted statistics in long format.
...	Not used.

Value

A data frame.

tidy.lmmr_pairwise *Tidy pairwise permutation tests*

Description

Tidy pairwise permutation tests

Usage

```
## S3 method for class 'lmmr_pairwise'
tidy(x, ...)
```

Arguments

x	An object of class lmmr_pairwise.
...	Not used.

Value

A data frame with one row per pair.

tidy.lmmr_perm *Tidy a permutation test*

Description

Tidy a permutation test

Usage

```
## S3 method for class 'lmmr_perm'
tidy(x, type = c("summary", "null"), ...)
```

Arguments

x	An object of class lmmr_perm.
type	"summary" returns one row with the test result; "null" returns the permutation distribution, one row per permutation.
...	Not used.

Value

A data frame.

`tidy.lmmr_timecourse` *Tidy a residual time-course diagnostic*

Description

Tidy a residual time-course diagnostic

Usage

```
## S3 method for class 'lmmr_timecourse'
tidy(x, ...)
```

Arguments

`x`	An object of class `lmmr_timecourse`.
`...`	Not used.

Value

A data frame with the mean residual and its standard error at each time point and group.

Index

boot_ci, 3
boot_ci(), *5*, *8*, *17*, *30*

confint(), *4*
confint.lmmr_boot, 5

diag_acf, 6
diag_acf(), *10*
diag_agreement, 7
diag_loso, 8
diag_loso(), *5*
diag_timecourse, 9
diag_timecourse(), *7*, *8*, *13*

exch_free *(*exchange*)*, 10
exch_free(), *12*, *19*, *21*
exch_signflip *(*exchange*)*, 10
exch_signflip(), *12*, *19–21*
exch_within *(*exchange*)*, 10
exch_within(), *12*, *19*, *21*
exchange, 10

future::plan(), *20*

ggplot2::ggplot(), *22–25*

lme4::bootMer(), *4*
lme4::lmer(), *3*, *6–9*, *12*, *16*, *18*, *19*

perm_calibrate, 11
perm_calibrate(), *6*, *7*, *11*, *21*
perm_maxt, 14
perm_maxt(), *17*, *18*
perm_pairwise, 16
perm_spec, 18
perm_spec(), *14*, *15*
perm_test, 19
perm_test(), *5*, *10–17*
plot(), *4*, *6–9*, *13*, *15*, *21*
plot.lmmr_acf, 22
plot.lmmr_agreement, 22
plot.lmmr_boot, 23
plot.lmmr_calib, 23
plot.lmmr_loso, 24
plot.lmmr_maxt, 24
plot.lmmr_perm, 25
plot.lmmr_timecourse, 25
print(), *4*, *6–9*, *13*, *15*, *17*, *21*
progressr::with_progress(), *20*

sim_blocks, 26
stats::confint(), *5*
stats::lm(), *3*, *6–9*, *12*, *16*, *18*, *19*
summary(), *21*

tidy(), *4*, *6–9*, *13*, *15*, *17*, *21*
tidy.lmmr_acf, 28
tidy.lmmr_agreement, 29
tidy.lmmr_boot, 30
tidy.lmmr_calib, 30
tidy.lmmr_loso, 31
tidy.lmmr_maxt, 31
tidy.lmmr_pairwise, 32
tidy.lmmr_perm, 32
tidy.lmmr_timecourse, 33