The New **lmerTest** Package

*Rune Haubo B Christensen on behalf of the **lmerTest** authors*

March 2018

The **lmerTest** package has been re-written and is released as package version ≥ 3.0. We call this *The New **lmerTest** Package*. This document describes the key changes in the new version.

The **lmerTest** package provides p-values in type I, II or III `anova` and `summary` tables for linear mixed models (`lmer` model fits cf. **lme4**) via Satterthwaite's degrees of freedom method; a Kenward-Roger method is also available via the **pbkrtest** package. Model selection and assessment methods include `step`, `drop1`, anova-like tables for random effects (`ranova`), least-square means (LS-means; `lsmeans`) and tests of linear contrasts of fixed effects (`contest`).

The most important changes for end-users are:

1. More robust and less error-prone implementation with much better test coverage
2. Faster evaluation of summary and anova tables - much faster for time-consuming model fits
3. New functions including `drop1()`, `contest()`, `as_lmerModLmerTest()`, `show_test()`, and `ranova()` – see details below.

Given that the codebase has been rewritten completely from scratch it is not unlikely that a few 'childhood diseases' are lurking in the details. Please help us cure out such maladies and report bugs if you see them at https://github.com/runehaubo/lmerTestR/issues.

The new user interface is almost 100% backward compatible with previous versions (see details below). An up-to-date version of this document is available here.

New features:

1. `ranova()` - ANOVA-like table of random effects via likelihood ratio tests with methods for both `lmerMod` and `lmerModLmerTest` objects. `ranova()` is similar to the old `rand()` and essentially produces a `drop1()` table for random-effect terms. `ranova()` can either test *reduction* of random-effect terms to simpler structures or it can test *removal* of entire random-effect terms; the rules for how complex random-effect terms are reduced is described in `help(ranova)`.
2. `drop1()` - F-tests of fixed-effect terms using Satterthwaite or Kenward-Roger methods for denominator degrees of freedom. These 'single term deletion' tables are useful for model selection and tests of marginal terms. Compared to the likelihood ratio tests of `lme4::drop1` the F-tests and p-values of `lmerTest::drop1` are more accurate and considerably faster since no additional model fitting is required.
3. `as_lmerModLmerTest()` - an explicit `coerce` function from class `'lmerMod'` to `'lmerModLmerTest'`.
4. `contest()` - tests of contrasts, i.e. tests of linear functions of the fixed-effect coefficients. A user-friendly interface for tests of contrasts with outputs either as a summary-like table of t-tests or an anova-like table of F-tests (or a list of either). Contrasts can optionally be tested for estimability. Contrasts are allowed to be rank-deficient as the rank is automatically detected and appropriate adjustments made. Methods for `'lmerModLmerTest'` as well as `'lmerMod'` objects – the latter avoids the Satterthwaite specific computations when the Kenward-Roger method is used.
5. A `show_test()` function which operates on anova tables and LS-means tables (produced by `ls_means`) makes it possible to see exactly which functions of the coefficients are being tested. This is very helpful when differences between type I, II and III anova tables are being considered and discussed.

6. An `ls_means` functions is provided as an alias for (the perhaps not so intuitively named) `lsmeansLT` function. As the name implies the function computes the so-called least-squares means (classical Yates contrasts) as well as pairwise differences of these.
7. `lmerTest::lmer` returns an object of class `'lmerModLmerTest'` (previously `'merModLmerTest'`) to clarify that `'lmerModLmerTest'` extends `'lmerMod'` – not `'merMod'`. The `merMod` class includes generalized and nonlinear mixed models and **lmerTest** is only designed for *linear* mixed models.
8. Test coverage has been greatly improved providing confidence that **lmerTest** functionality works as expected even in boundary situations (e.g. such that anova and summary tables have the expected format even if there are no fixed effects - and even if the intercept has been suppressed as well.)
9. The computational approach is to let `lmerTest::lmer` compute the required Hessian and derivatives needed for evaluation of degrees of freedom and t- and F-tests. Previously these quantities were computed following calls to `anova` and `summary` methods which meant that the computationally intensive parts had to be evaluated anew with each `summary` or `anova` table; the model even had to be refitted as well. With the new implementation refitting the model is avoided and the required Hessian and derivaties have to be computed only once per model fit.
10. The number of dependent packages has been reduced easing the installation of **lmerTest**.
11. Consistency of output - user visible functions as well as internal functions take care to return objects of the appropriate form even in boundary situations, e.g. always a matrix or always a list.

Changes to the user interface

The user interface has been updated with new functionality and extra features as described above. We have tried our best to keep the new version backward compatible, but a few things from the old API have been changed out of necessity:

1. In `step()` the argument `type` is now being ignored as `drop1()` is always used for reduction of the fixed-effect structure. `type` used to indicate the type of anova table to use for tests of fixed-effect terms, but since it only makes sense to remove marginal terms and `drop1()` provides the test of these terms it does not make sense to use anything but `drop1()` to test the fixed-effect terms. Furthermore type II and III anova tables provide identical tests of marginal terms. Additionally, the arguments `fixed.calc`, `lsmeans.calc`, `difflsmeans.calc`, and `test.effs` are deprecated and attempts to set them leads to a warning; also `keep.effs` has been reduced to `keep`.
2. The documented behavior of `rand` (which is based on `ranova`) has not changed, but being a new implementation it may behave differently in 'corner' cases.
3. `anova` tables have a column `F value` (previously `F.value`) being consistent with `anova.lm` and `anova.merMod`.
4. The headers for `summary` and `anova` tables have been modernized.

1. Plot methods for `step` and `lsmeans` objects have been rewritten and may behave differently. These plot methods are mostly provided for backward compatibility. In general we recommend plot methods provided by the **emmeans** package for plots if LS-means.

Plans for future releases:

1. `calcSatterth()` will be `.Deprecated` and eventually `.Defunct` since its functionality is covered by the new function `contest()`.
2. `rand()` is an alias for the new function `ranova()`. `rand` may be `.Deprecated` in future releases.

3. Plot methods for `ls_means` and `step` objects may be discontinued (and they have never been documented and part of the public API) since much better alternatives are available in the **emmeans** package.

Changes relevant for programmers and downstream packages

1. `lmerTest::lmer` produces an object of (`S4`) class `'lmerModLmerTest'` (previously `'merModLmerTest'`) which extends the `'lmerMod'` class - objects of class `'lmerMod'` are produced by `lme4::lmer`.
2. `anova` and `summary` methods for objects of class `'lmerModLmerTest'` are `S3` and should only be called using method dispatch, i.e. they should be called with objects of class `'lmerModLmerTest'` as the first argument. If you have an `object` of class `'lmerMod'` and want to compute p-values use `anova(as_lmerModLmerTest(object))` or `summary(as_lmerModLmerTest(object))`. (The previous **lmerTest** package defined `S4` `anova` and `summary` methods.)
3. `anova` and `summary` methods are not exported functions and they are not designed to be called with `lmerTest::summary` and `lmerTest::anova`. Use `anova(as_lmerModLmerTest(object))` or `summary(as_lmerModLmerTest(object))` instead if `object` is of class `lmerMod`, i.e. the result of calling `lme4::lmer`.
4. If your package *suggests* (rather than imports or depends on) **lmerTest**, the canonical way to enforce calls to the `anova` and `summary` methods defined by **lmerTest** is as follows:

```r
# For packageVersion("lmerTest") >= "3.0.0" :
if(requireNamespace("lmerTest", quietly = TRUE)) {
  # for summary() change anova -> summary here:
  anova(lmerTest::as_lmerModLmerTest(object)) # optionally add add. args.
} else stop("Package lmerTest is not available.")
```

Bridging between versions for package programmers

1. If you want to test if an `object` is of class `'lmerModLmerTest'` *or* `'merModLmerTest'`, use `inherits(object, "merModLmerTest") || inherits(object, "lmerModLmerTest")`: this tests `TRUE` for both classes.
2. If you are working with `anova` and `summary` methods for `lmer` objects (`lme4::lmer` or `lmerTest::lmer`) in your package, and you want to make sure to call the `anova` and `summary` methods defined by **lmerTest** you may use the following functions to transition from the old to the new **lmerTest** package.

```r
lmerTest_anova <- function(object, ...) {
  # Produce lmerTest-anova table for lmer-model fits (lme4 or lmerTest) with old
  # as well as new lmerTest package.
  # Standard method dispatch for all non-lmerMod objects.
  if(!inherits(object, "lmerMod")) return(anova(object, ...)) # non-lmer objects
  if(requireNamespace("lmerTest", quietly=TRUE) && packageVersion("lmerTest") < "3.0.0") {
    if(inherits(object, "merModLmerTest"))
      return(lmerTest::anova(object, ...))  else # lmerTest object
        return(lmerTest::anova(as(object, "merModLmerTest"), ...)) # lme4 object
  }
  if(requireNamespace("lmerTest", quietly=TRUE) && packageVersion("lmerTest") >= "3.0.0") {
    if(inherits(object, "lmerModLmerTest"))
      return(anova(object, ...)) else # lmerTest object
        return(anova(lmerTest::as_lmerModLmerTest(object), ...)) # lme4 object
  }
  return(anova(object, ...)) # *merModLmerTest objects and/or 'lmerTest' is not available
}
```

```r
lmerTest_summary <- function(object, ...) {
  # Produce lmerTest-summary for lmer-model fits (lme4 or lmerTest) with old
  # as well as new lmerTest package.
  # Standard method dispatch for all non-lmerMod objects.
  if(!inherits(object, "lmerMod")) return(summary(object, ...)) # non-lmer objects
  if(requireNamespace("lmerTest", quietly=TRUE) && packageVersion("lmerTest") < "3.0.0") {
    if(inherits(object, "merModLmerTest"))
      return(lmerTest::summary(object, ...))  else # lmerTest object
        return(lmerTest::summary(as(object, "merModLmerTest"), ...)) # lme4 object
  }
  if(requireNamespace("lmerTest", quietly=TRUE) && packageVersion("lmerTest") >= "3.0.0") {
    if(inherits(object, "lmerModLmerTest"))
      return(summary(object, ...)) else # lmerTest object
        return(summary(lmerTest::as_lmerModLmerTest(object), ...)) # lme4 object
  }
  return(summary(object, ...)) # *merModLmerTest objects and/or 'lmerTest' is not available
}
```