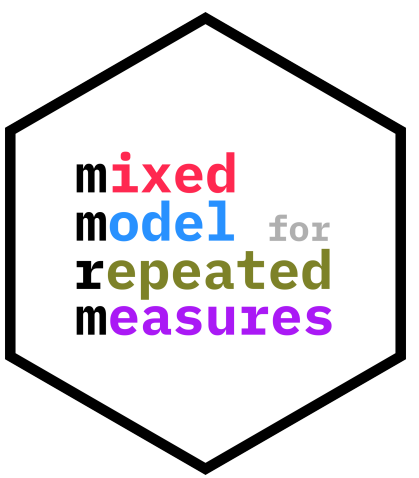
{mmrm}: A Robust and Comprehensive R Package for Implementing Mixed Models for Repeated Measures

useR! 2024

Daniel Sabanés Bové
RCONIS

July 9, 2024



Acknowledgments

Thanks to all other authors of `{mmrm}`:

- Brian Matthew Lang (MSD)
- Christian Stock (Boehringer)
- Dan James (AstraZeneca)
- Daniel Leibovitz (Roche)
- Daniel Sjoberg (Roche)
- Doug Kelkhoff (Roche)
- Julia Dedic (Roche)
- Jonathan Sidi (Sanofi)
- Kevin Kunzmann (Boehringer)
- Liming Li (Roche)
- Ya Wang (Gilead)

Acknowledgments

Thanks for discussions and contributions from:

- Ben Bolker (McMaster University)
- Davide Garolini (Roche)
- Craig Gower-Page (Roche)
- Dinakar Kulkarni (Roche)
- Gonzalo Duran Pacheco (Roche)
- Members of `openstatsware`

Agenda

- Prelude: `openstatsware`
- Interlude: Mixed Models for Repeated Measures and `{mmrm}`
- Finale: Ingredients for Successful Collaborations

Prelude: openstatsware

openstatsware

- Formed on 19 August 2022, affiliated with American Statistical Association (ASA) as well as European Federation of Statisticians in the Pharma Industry (EFSPI)
- Cross pharma industry collaboration (56 members from 35 organizations)
- Homepage at openstatsware.org
- We welcome new members to join!

Working Group Objectives

- Primary
  - Engineer R packages that implement important statistical methods
    - to fill in gaps in the open-source statistical software landscape
    - focusing on what is needed for biopharmaceutical applications
- Secondary
  - Develop and disseminate best practices for engineering high-quality open-source statistical software
    - By actively doing the statistical engineering work together, we align on best practices and can communicate these to others
    - See my virtual presentation about `openstatsguide` and the corresponding poster tomorrow!

Interlude: Mixed Models for Repeated Measures and {mmrm}

What is a MMRM?

- MMRM is a popular choice for analyzing longitudinal continuous outcomes in randomized clinical trials
- For each subject i we observe a vector

$$Y_i = (y_{i1}, \ldots, y_{im_i})^\top \in \mathbb{R}^{m_i}$$

- Use a design matrix $X_i \in \mathbb{R}^{m_i \times p}$
- Use a corresponding coefficient vector $\beta \in \mathbb{R}^p$
- Assume that the observations are multivariate normal distributed:

$$Y_i \sim N(X_i\beta, \Sigma_i)$$

  where the covariance matrix $\Sigma_i \in \mathbb{R}^{m_i \times m_i}$ is derived by subsetting the overall covariance matrix $\Sigma \in \mathbb{R}^{m \times m}$ appropriately

Covariance model and Estimation

- The symmetric and positive definite covariance matrix Σ is parametrized by a vector of variance parameters $\theta = (\theta_1, \ldots, \theta_k)^\top$
- There are many different choices for how to model the covariance matrix and correspondingly θ has different interpretations, e.g.:
  - Unstructured, Toeplitz, AR1, compound symmetry, ante-dependence, spatial exponential
  - Group specific covariance estimates and weights
- Estimation is performed
  - (sometimes) via maximum likelihood (ML) inference, maximizing the joint log-likelihood of (β, θ) or
  - (usually) via integrating out β from the likelihood and maximizing for θ (restricted ML, REML)

Existing R Packages and Tried Solutions

- One challenge is that we need to use “adjusted” degrees of freedom for t- or F-test statistics
  - because of the covariance parameter estimation and data sets are usually unbalanced
  - Typical Satterthwaite or Kenward-Roger adjustment methods
- Initially thought that the MMRM problem was solved by using `lme4` with `lmerTest`, learned that this approach failed on large data sets (slow, did not converge)
- `nlme` does not give Satterthwaite adjusted degrees of freedom, has convergence issues, and with `emmeans` it is only approximate
- Next we tried to extend `glmmTMB` to calculate Satterthwaite adjusted degrees of freedom, but it did not work

New Idea

- We only want to fit a fixed effects model with a structured covariance matrix for each subject
- The idea is then to use the Template Model Builder (`TMB`) directly
  - as it is also underlying `glmmTMB`
  - but code the exact model we want
- We do this by implementing the log-likelihood in `C++` using the `TMB` provided libraries
- Provide an R solution that
  - has fast convergence times
  - generates estimates closest to (previous) “gold standard” implementation (`SAS`)

Advantages of TMB

- Fast C++ framework for defining objective functions (Rcpp would have been alternative interface)
- Automatic differentiation of the log-likelihood as a function of the variance parameters
- We get the gradient and Hessian exactly and without additional coding
- Syntactic sugars to allow simple matrix calculations or operations like R
- This can be used from the R side with the TMB interface and plugged into optimizers

Why it's not just another package

- Ongoing maintenance and support from the pharmaceutical industry
  - 5 companies being involved in the funding, on track to become standard package
- Development using best practices as show case for high quality package
  - Thorough and transparent unit and integration tests to ensure accurate results
  - The integration tests in `{mmrm}` are set to a tolerance of 10^{-3} when compared to SAS outputs.
  - Uses the `testthat` framework with `covr` to communicate the testing coverage

Highlighted Features of `mmrm`

- **Hypothesis Testing**:
  - `emmeans` interface for least square means
  - Satterthwaite and Kenward-Roger adjustments
  - Robust sandwich estimator for covariance
- **Integrations and extentions**
  - `tidymodels` builtin parsnip engine and recipes for streamlined model fitting workflows
  - `teal`, `tern`, `rtables` integration for post processing and reporting
  - Support conditional mean prediction and simulation
  - Also used in `rbmi` for conditional mean imputation!

Computational Efficiency

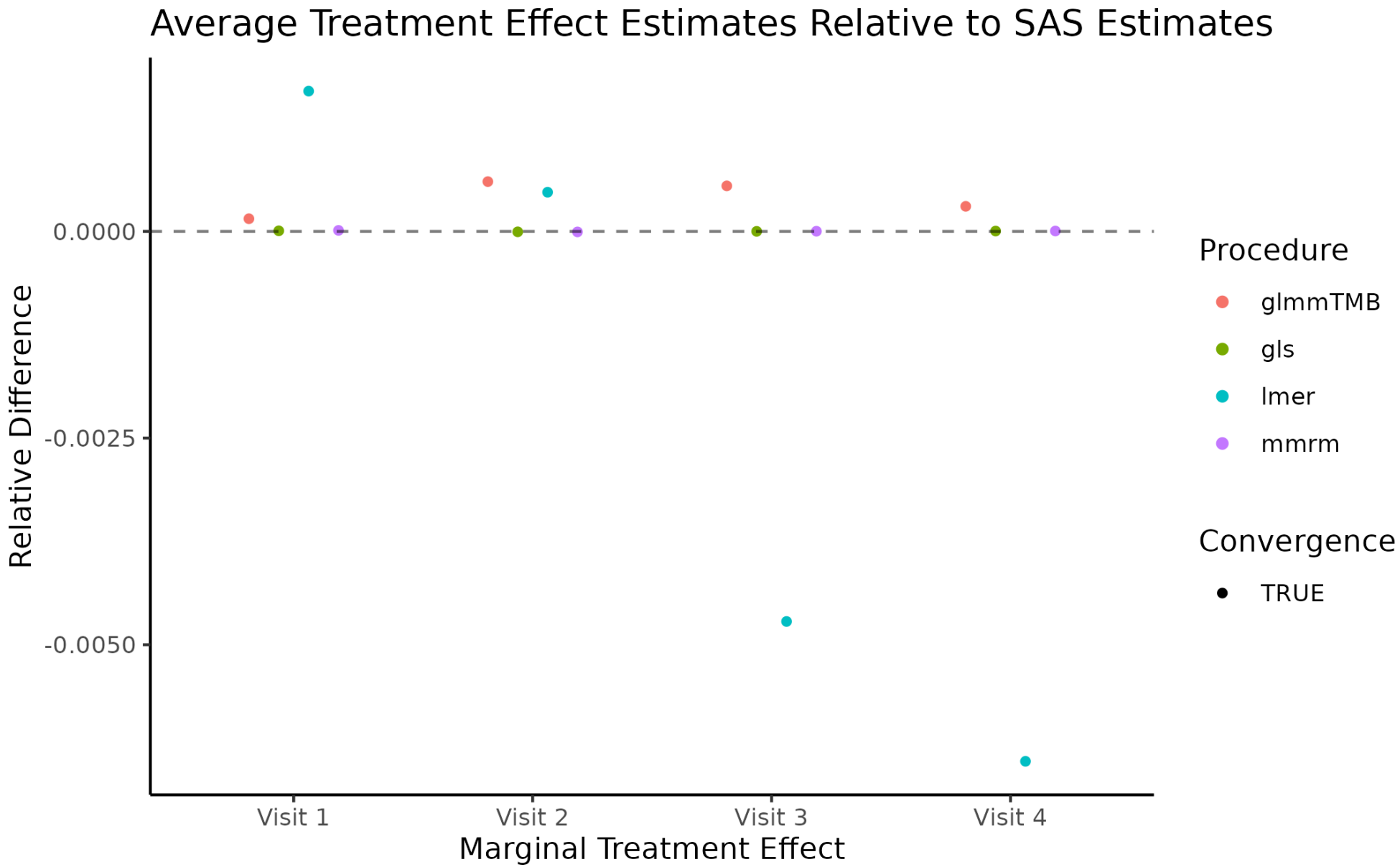
`mmrm` not only supports multiple covariance structure, it also has good efficiency (due to fast implementations in C++)

Comparison of convergence times (ms) for example (source)

Implementation	Median	First Quartile	Third Quartile
`mmrm`	56.15	55.76	56.30
`PROC GLIMMIX`	100.00	100.00	100.00
`lmer`	247.02	245.25	257.46
`gls`	687.63	683.50	692.45
`glmmTMB`	715.90	708.70	721.57

Differences with SAS

`{mmrm}` has small difference from SAS



Marginal mean treatment effects for each visit (source)

Impact of `mmrm`

- CRAN downloads: 3922 per month in the last month
  - https://cran.r-project.org/web/packages/mmrm/
- GitHub repository: 101 stars as of 4th July 2024
  - https://github.com/openpharma/mmrm
- Quite a lot of questions on StackOverflow (and internal similar question boards)
- Most important features have been implemented by now, but definitely open for feature requests and grateful for any bug reports!

Getting started

- `mmrm` is on CRAN - use this as a starting point:

```r
install.packages("mmrm")
library(mmrm)
fit <- mmrm(
  formula = FEV1 ~ RACE + SEX + ARMCD * AVISIT + us(AVISIT | USUBJID),
  data = fev_data
)
summary(fit)
library(emmeans)
emmeans(fit, ~ ARMCD | AVISIT)
```

- Visit openpharma.github.io/mmrm for detailed docs including vignettes
  - In particular the comparison vignette
- Consider tern.mmrm for high-level clinical reporting interface, incl. standard tables and graphs

Finale: Ingredients for successful and sustainable collaboration

Human factors

- Mutual interest and trust
- Prerequisite is getting to know each other
  - Although mostly just online, biweekly calls help a lot with this
- Reciprocity mindset
  - “Reciprocity means that in response to friendly actions, people are frequently much nicer and much more cooperative than predicted by the self-interest model”
  - Personal experience: If you first give away something, more will come back to you.

Development process

- Important to go public as soon as possible
  - don’t wait for the product to be finished
  - you never know who else might be interested/could help
- Version control with git
  - cornerstone of effective collaboration
- Building software together works better than alone
  - Different perspectives in discussions and code review help to optimize the user interface and thus experience

Coding standards

- Consistent and readable code style simplifies joint work
- Written (!) contribution guidelines help
- Lowering the entry hurdle using developer calls is important

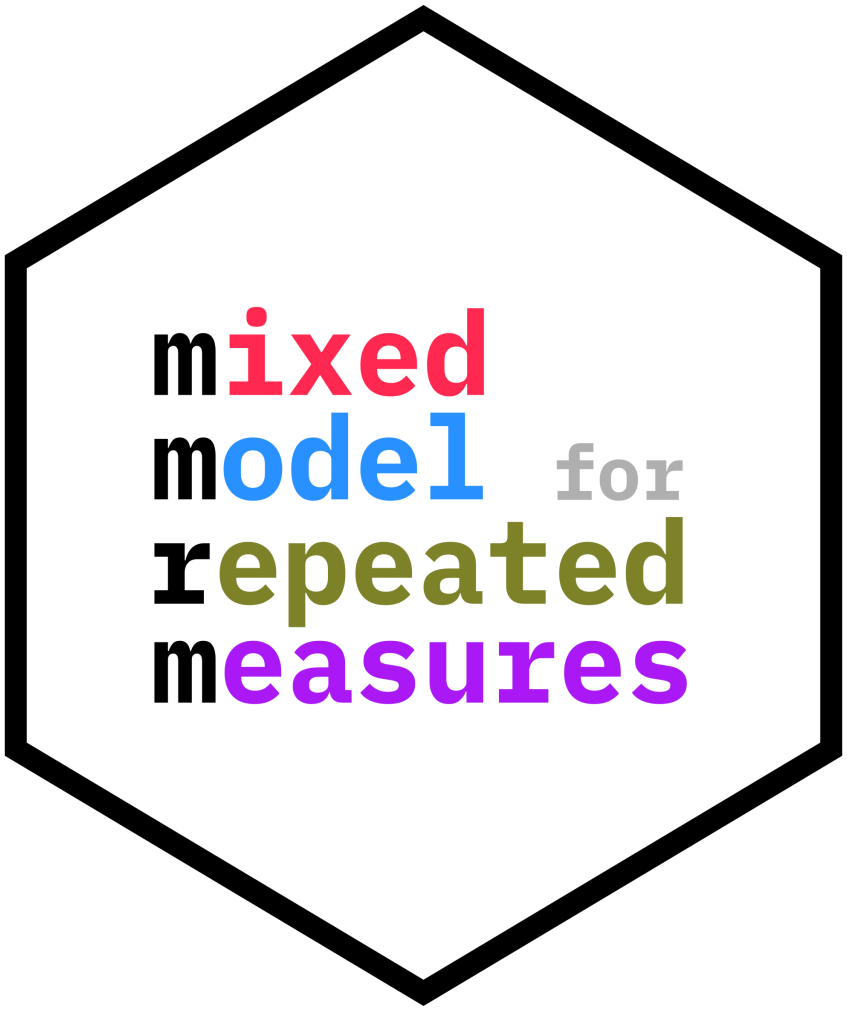
Robust test suite

- Unit and integration tests are essential for preventing regression and assuring quality
- Especially with compiled code critical to see if package works correctly
- Use continuous integration during development to make sure nothing breaks along the way

Documentation

- Lots of work but extremely important
  - start with writing up the methods details
  - think about the code structure first in a “design doc”
  - only then put the code in the package
- Needs to be kept up-to-date
- Need to have examples & vignettes
  - Testing alone is not sufficient
  - Builds trust with users
  - Reference for developers over time

Thank you! Questions?



0 reactions

0 comments