Week 11: Practical lme4 workflow + convergence

Warnings, singular fits, scaling, optimizers, and principled simplification

Bartosz Maćkiewicz

Why this week exists

Mixed models are powerful, but they are also **easy to misuse**:

- models can be too complex for the data
- optimizers can struggle (convergence warnings)
- random-effects estimates can hit boundaries (singular fits)

This week is about a **repeatable workflow** for diagnosing and fixing problems without “p-hacking your random effects”.

The goal (stated plainly)

When `lmer()` complains, you have two jobs:

1. **Decide whether the complaint matters** for your scientific question.
2. If it does matter, **fix the model in a defensible way** (not by trial-and-error until the warning disappears).

The point is not “make the warning go away.”

The point is: fit a model that is plausible for your design and produces estimates you are willing to interpret and report.

The 3 most common problems

1) **Convergence warnings**

Typical message:

```
Model failed to converge with max|grad| = ... (tol = ...)
```

2) **Singular (boundary) fits**

Typical message:

```
boundary (singular) fit: see help('isSingular')
```

3) **Degenerate / unstable covariance estimates**

Symptoms:

- random-effect variances near zero
- random-effect correlations near ±1
- warnings about Hessian / negative eigenvalues

What “convergence” means (in practice)

`lmer()` estimates:

- fixed effects (the population-level coefficients)
- random effects variance-covariance parameters
- residual variance

Convergence warnings often mean: **the optimizer is not confident it found a stable optimum**.

Sometimes the warning is harmless; sometimes it indicates your model is not trustworthy.

What not to do

When you see a warning, do not:

- remove random effects purely because it changes p-values
- keep retrying random optimizers until the warning disappears
- report the model without mentioning that it struggled

All of those are versions of “silent researcher degrees of freedom.”

Instead: you follow a workflow.

First response: do not guess

When you see warnings, do this first:

1) **Write down the model** (fixed effects + random effects + coding decisions).
2) Check:
    - Did you scale/center predictors?
    - Do you have enough data per grouping level?
    - Are there near-zero variances or near-perfect correlations?
3) Use diagnostic helpers:

```
performance::check_convergence(fit)
performance::check_singularity(fit)
isSingular(fit)
VarCorr(fit)
```

A repeatable workflow (the version you should memorize)

1. Start with a model you can justify from the **design**.
2. Check model behavior (warnings + `check_*` helpers).
3. If there are problems, apply fixes in a defensible order:
    - scaling/centering (interpretation + numerics)
    - reduce correlation parameters (`||`)
    - simplify slopes *based on design and information in the data*
    - only then: optimizer tweaks
4. Report what you tried and why.

Worked example: `sleepstudy` (built-in)

We reuse a familiar dataset so we can focus on the workflow.

We will fit:

- a random intercept model
- a random intercept + random slope model

Then we will check convergence/singularity and interpret what those diagnostics mean.

```
sleep <- as_tibble(sleepstudy) |>
  mutate(Days = Days - 2) |>
  filter(Days >= 0)

sleep |> count(Subject)
```

```
# A tibble: 18 x 2
   Subject     n
   <fct>   <int>
```

Fit a reasonable baseline model (random intercepts)

```
fit_ri <- lmer(Reaction ~ Days + (1 | Subject), data = sleep)
fit_ri
```

```
Linear mixed model fit by REML ['lmerModLmerTest']
Formula: Reaction ~ Days + (1 | Subject)
   Data: sleep
REML criterion at convergence: 1430.02
Random effects:
 Groups   Name        Std.Dev.
 Subject  (Intercept) 41.80
 Residual             30.22
Number of obs: 144, groups:  Subject, 18
Fixed Effects:
(Intercept)         Days
     267.97        11.44
```

This is a good first “working” model because:

Check convergence and singularity (baseline)

```
performance::check_convergence(fit_ri)
```

```
[1] TRUE
attr(,"gradient")
[1] 2.433154e-09
```

```
performance::check_singularity(fit_ri)
```

```
[1] FALSE
```

```
VarCorr(fit_ri)
```

```
 Groups   Name        Std.Dev.
 Subject  (Intercept) 41.796
 Residual             30.217
```

If the model is well-behaved, you should see:

- no convergence problems flagged
- no singularity flagged

Add a random slope (more realistic, more complex)

Allow each subject to have their own trajectory over days:

```
fit_rs <- lmer(Reaction ~ Days + (Days | Subject), data = sleep)
fit_rs
```

```
Linear mixed model fit by REML ['lmerModLmerTest']
Formula: Reaction ~ Days + (Days | Subject)
   Data: sleep
REML criterion at convergence: 1404.094
Random effects:
 Groups   Name        Std.Dev. Corr
 Subject  (Intercept) 30.957
          Days         6.766   0.18
 Residual             25.526
Number of obs: 144, groups:  Subject, 18
Fixed Effects:
(Intercept)         Days
     267.97        11.44
```

Check convergence and singularity (random slopes)

```
performance::check_convergence(fit_rs)
```

```
[1] TRUE
attr(,"gradient")
[1] 7.320904e-06
```

```
performance::check_singularity(fit_rs)
```

```
[1] FALSE
```

```
VarCorr(fit_rs)
```

```
 Groups   Name        Std.Dev. Corr
 Subject  (Intercept) 30.9573
          Days         6.7659  0.178
 Residual             25.5264
```

Two important points:

1. A warning is not automatically fatal, but you do not ignore it.

What a “singular fit” means

When `lme4` says “boundary (singular) fit,” it usually means:

- at least one variance component is estimated (very) close to zero, **or**
- the random-effects covariance matrix is not full rank (some parameters are not identifiable from the data)

This can happen because:

- the true variance is near zero (so the model is too ambitious)
- the dataset is too small / not informative for that structure
- the random-effects structure is redundant for the design

Singularity is often a design/information issue

A common misconception is that singularity is “a numerical error.”

Usually it is information:

- You asked the model to estimate more random-effect parameters than your data can support.
- The optimizer then does the only honest thing it can do: it pushes some parameters to the boundary.

So the best fix is usually not “different optimizer,” but “different structure.”

Scaling and centering (cheap and often fixes issues)

Problems are more likely when predictors are on very different scales.

Rule of thumb:

- **center** predictors when 0 is not meaningful
- **scale** predictors (z-score) when magnitudes differ

```
df <- df |>
  mutate(
    x_z = as.numeric(scale(x)),
    z_z = as.numeric(scale(z))
  )
```

Why scaling helps (not just numerics)

Scaling/centering also helps interpretability:

- If 0 is not meaningful, the intercept becomes meaningless and can be poorly estimated.
- Centering makes the intercept correspond to a meaningful “typical” value of predictors.
- Scaling makes coefficients comparable (in the same units of SD change), which can be useful in exploratory modeling.

So scaling/centering is often the right first step even when the model technically converges.

Random slopes increase complexity fast

Each random slope adds:

- a variance parameter
- (often) covariances with other random effects

Common symptoms of “too complex”:

- max|grad| convergence warnings
- correlations close to ±1
- singular fits

The goal is not maximal complexity; it is **a defensible structure supported by the design and data**.

| vs || (a very practical trick)

`(x | group)` estimates:

- random intercept variance
- random slope variance
- their correlation

`(x || group)` estimates the variances but **forces the correlation to 0**.

Why this helps:

- correlations are often weakly identified unless you have lots of groups and lots of observations per group
- a near ±1 correlation is a common symptom of overparameterization

So || can be a principled simplification step: it reduces complexity without changing which slopes are allowed to vary.

A principled simplification ladder

If the model is unstable, simplify in a defensible order:

1) Scale/center predictors.
2) Remove correlation parameters using `||`:

```
(x + z || group)
```

3) Drop the least-justified random slopes (keep the design in mind).
4) If needed, move to a simpler random-effects structure (random intercept only).

Document your decision: what you changed and why.

Optimizers and controls (when simplification is not enough)

Try a different optimizer and/or more iterations:

```
ctrl <- lmerControl(
  optimizer = "bobyqa",
  optCtrl = list(maxfun = 2e5)
)

fit2 <- update(fit1, control = ctrl)
```

If changing the optimizer “fixes” the warning but results are unstable, it may still indicate overfitting.

How to tell if an optimizer change is meaningful

If you change the optimizer and “the warning disappears,” you still check:

- Are parameter estimates similar across optimizers?
- Are standard errors and inferences stable?
- Does `VarCorr()` look reasonable (no extreme correlations, no weird near-zero variances that flip around)?

An optimizer change should not be treated as a license to ignore a model that is too complex for the data.

Model comparison: ML vs REML

Practical rule (course workflow):

- Compare **fixed effects** using **ML** (`REML = FALSE`).
- Refit the chosen model with **REML** for final variance estimates (`REML = TRUE`).

```
fit_small <- lmer(y ~ x + (1|g), data=df, REML=FALSE)
fit_big   <- lmer(y ~ x + z + (1|g), data=df, REML=FALSE)
anova(fit_small, fit_big)
```

Reporting “convergence work”

In your report (or appendix), state:

- the random-effects structure you attempted
- whether you observed warnings / singular fits
- what you changed (scaling, `||`, simplified slopes, optimizer)
- the final structure and why it is defensible

Avoid “silent” fixes.

A reporting template (copy/paste quality)

1. We initially fit `y ~ ... + (random structure)` based on the design.
2. We observed: (i) convergence warning / (ii) singular fit / (iii) none.
3. We then applied the following defensible adjustments (in order):
    - scaling/centering: ...
    - correlation removal with `||`: ...
    - random-slope simplification: ...
    - optimizer control: ...
4. The final model was: `...`
5. We justify it because: (design reason) + (data support reason).

Takeaways

- Convergence warnings are information, not noise.
- Start with scaling/centering and design-based simplification.
- Use `check_convergence()` / `isSingular()` and inspect `VarCorr()`.
- Use optimizers as a tool, not as a way to force a bad model to behave.