Reporting regression results

Extracting model estimates and data structure

Daniela Palleschi

2024-07-09

Table of contents

Learning objectives

Today we will...

- run our first linear (mixed) model
- produce model summary tables
- report analyses and cite packages
- report model results reproducibly

Resources

Set-up

Packages

- we'll need the following packages for today

```
pacman::p_load(
  tidyverse,
  lme4,
  broom,
  broom.mixed,
  brms
)
```

Data

- we'll need the dataset we've been working with

```
df_lifetime <- readr::read_csv(here::here("data", "tidy_data_lifetime_pilot.csv"),
                               # for special characters
                               locale = readr::locale(encoding = "latin1")
                               ) |>
  mutate_if(is.character,as.factor) |> # all character variables as factor
  filter(type == "critical", # only critical trials
         px != "px3") # this participant had lots of 0's for some reason
```

Data dictionary

- let's review the data dictionary

```
dict_lifetime <- read_csv(
  here::here("data", "tidy_data_lifetime_pilot_dictionary.csv")
)
```

Model prep

- we want to only have critical items

```
df_lifetime <-
  df_lifetime |>
  filter(type == "critical") |>
  droplevels()
```

- we want our predictors (`lifetime` and `tense`) to be factors

```
df_lifetime$lifetime <- as.factor(df_lifetime$lifetime)
df_lifetime$tense <- as.factor(df_lifetime$tense)
```

- and to have sum contrast coding:
  - `PP` and `living` are `-0,5`
  - `SF` and `dead` are `+0,5`

```
contrasts(df_lifetime$lifetime)
```

```
       living
dead        0
living      1
```

```
contrasts(df_lifetime$tense)
```

```
   SF
PP  0
SF  1
```

- we see the default treatment (or 'dummy') contrasts here
- to change them:

```
contrasts(df_lifetime$lifetime) <- c(-0.5, +0.5)
contrasts(df_lifetime$tense) <- c(+0.5, -0.5)
```

```
contrasts(df_lifetime$lifetime)
```

```
       [,1]
dead   -0.5
living  0.5
```

```
contrasts(df_lifetime$tense)
```

```
   [,1]
PP  0.5
SF -0.5
```

1 Linear regression

- fitting a straight line to data
- for an introduction see webbook for my course Regression for Linguists, e.g., Ch. 1 Understanding straight lines
- fixed-effects only multiple regression:

```
lm(
  log(fp) ~ tense * lifetime,
  subset = region == "verb" & ff > 0,
  data = df_lifetime
)
```

```
Call:
lm(formula = log(fp) ~ tense * lifetime, data = df_lifetime,
    subset = region == "verb" & ff > 0)

Coefficients:
     (Intercept)            tense1         lifetime1  tense1:lifetime1
         5.63477           0.03357           -0.10130           -0.08320
```

1.1 Naming conventions

- it's very helpful to save our models to our environment so we can inspect them
  - always try to use a prefix that defines what the object is
  - e.g., `fit_` or `mod_` for models more generally
  - or `lm_` or `glm_` to specify the function used to produce the model
  - whatever you choose, be consistent within your project

```
lm_fp <-
  lm(
  log(fp) ~ tense * lifetime,
  subset = region == "verb" & ff > 0,
  data = df_lifetime
)
```

1.2 Model summary

- the most straightforward way to inspect your model and the results is with `summary()`

```
summary(lm_fp)
```

```
Call:
lm(formula = log(fp) ~ tense * lifetime, data = df_lifetime,
    subset = region == "verb" & ff > 0)

Residuals:
     Min       1Q   Median       3Q      Max
-1.18141 -0.34282  0.00058  0.26923  1.38822

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       5.63477    0.01877 300.193  < 2e-16 ***
tense1            0.03357    0.03754   0.894  0.37158
lifetime1        -0.10130    0.03754  -2.698  0.00718 **
tense1:lifetime1 -0.08320    0.07508  -1.108  0.26828
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4374 on 539 degrees of freedom
Multiple R-squared:  0.01712,   Adjusted R-squared:  0.01165
F-statistic: 3.129 on 3 and 539 DF,  p-value: 0.02538
```

1.3 Extracting information

- `lme4` objects contains lots of information
  - navigate to the model object in your Environment and explore the elements
- for example
  - `nobs(model)` will tell you how many data points were included in the model
  - `formula(model)` will give you the model formula

```
nobs(lm_fp)
```

```
[1] 543
```

```
formula(lm_fp)
```

```
log(fp) ~ tense * lifetime
```

- want to check which contrasts were used in your model?

```
lm_fp$contrasts
```

```
$tense
   [,1]
PP  0.5
SF -0.5

$lifetime
       [,1]
dead   -0.5
living  0.5
```

2 Running models reproducibly

- as your models get more computationally complex, they can take time to fit
  - sometimes you also run several models in the same script, which can take a long time
- with literate programming, we want to be able to produce dynamic reports
  - we don't want to run our models anew every time we re-render our document!
  - this also might result in different results if e.g., we update a package or R version
  - or if we send the script to somebody else with a different set-up
- ideally, we would have our source code (in our `.qmd` script), output (results printed in e.g., HTML our PDF output), *and* the actual model all stored and retrievable (`.rds` files)

2.1 Storing our models

- we can save our models as R Data Serialisation files (RDS, `.rds` file extension)
  - `saveRDS(object, filename)` writes an R object as RDS file
  - `readRDS(filename)` reads it in

- first, you'd want to create a folder where you store your models
  - then save your models once you've run them
  - but MAKE SURE you set `eval: false`
  - or even better, comment out this code!

````
```{r}
#| eval: false
saveRDS(lm_fp, here::here("models", "lm_fp"))
```
````

- ideally you would also set the code chunks that fit the model to `echo: false`
  - you don't want to re-fit the model every time you render the document
  - what's more, any differences in the re-fit model will not reflect what you're actually reporting elsewhere when using the stored RDS model

2.2 Load in RDS model

- and load them in e.g., when you want to report your models in Rmarkdown

```
lm_fp <- readRDS(here::here("models", "lm_fp"))
```

3 Model reporting

- when you run your models, you'll want to also print the output
  - this way you avoid needing to re-run your code to see the results
- for `lme4` models we can use `summary()`
  - as well as `formula()`, `nobs()`

3.1 `broom`

- the `broom` package also contains useful functions for reporting `lme4` models
  - `broom::tidy()` produces the model estimates for our fixed effects

```
broom::tidy(lm_fp)
```

Table 1: Example model summary table using `broom::tidy()`, `knitr::kable()`, and `kableExtra::kable_styling()`

term	estimate	std.error	statistic	p.value
(Intercept)	5.63	0.02	300.19	$< .001$
tense1	0.03	0.04	0.89	0.372
lifetime1	-0.10	0.04	-2.70	0.007
tense1:lifetime1	-0.08	0.08	-1.11	0.268

```
# A tibble: 4 x 5
  term             estimate std.error statistic p.value
  <chr>               <dbl>     <dbl>     <dbl>   <dbl>
1 (Intercept)        5.63      0.0188   300.     0
2 tense1             0.0336    0.0375     0.894 0.372
3 lifetime1         -0.101     0.0375    -2.70  0.00718
4 tense1:lifetime1  -0.0832    0.0751    -1.11  0.268
```

- it's also helpful to store the output as an object

```
tidy_lm_fp <- broom::tidy(lm_fp)
```

3.1.1 Example table

```
tidy_lm_fp |>
  mutate(p.value = case_when(
    p.value < .001 ~ "< .001",
    TRUE ~ as.character(round(p.value,3))
  )) |>
  knitr::kable("latex",
               digits = 2) |>
  kableExtra::kable_styling()
```

3.2 Publication-ready tables

- we've already seen how to produce publication-ready tables
  - TASK: produce a summary of the fixed effects of a model and feed it into our `knitr` and `kableExtra` functions to produce a publication-ready table

4 Mixed models

- the assumption of independence is often violated
  - due to multiple observations collected from e.g., the same person or item, language, etc.
- this can lead to the inflation of Type I error (the chance of a false positive)
- to account for this nonindependence of data points, we can used *mixed* models
  - *mixed* because they contain fixed effects (i.e., our predictors at the population-level) as well as *random* effects (group-level overall means and predictor effects)

4.1 `lmer()`

```
lmer_lifetime <-
  lmer(log(fp)  ~ tense * lifetime +
         (1|px) + (1|item_id),
       subset = region == "verb" & fp > 0,
       data = df_lifetime)
```

4.2 Extracting model elements

- we can use the same functions as for `lm()` models

```
nobs(lmer_lifetime)
```

```
[1] 543
```

```
formula(lmer_lifetime)
```

```
log(fp) ~ tense * lifetime + (1 | px) + (1 | item_id)
```

- we can also check how many levels we had for each grouping factor (i.e., random effect: items and participants)

```
summary(lmer_lifetime)$ngrps["px"]
```

```
px
 7
```

```r
summary(lmer_lifetime)$ngrps["item_id"]
```

```
item_id
     80
```

4.3 summary()

```r
summary(lmer_lifetime)
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: log(fp) ~ tense * lifetime + (1 | px) + (1 | item_id)
   Data: df_lifetime
 Subset: region == "verb" & fp > 0

REML criterion at convergence: 582.5

Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.99792 -0.67159  0.02895  0.67995  2.67790

Random effects:
 Groups   Name        Variance Std.Dev.
 item_id  (Intercept) 0.007012 0.08374
 px       (Intercept) 0.034124 0.18473
 Residual             0.154885 0.39355
Number of obs: 543, groups:  item_id, 80; px, 7

Fixed effects:
                 Estimate Std. Error t value
(Intercept)       5.63161    0.07245  77.736
tense1            0.03794    0.03389   1.120
lifetime1        -0.10308    0.03387  -3.044
tense1:lifetime1 -0.09587    0.06778  -1.414

Correlation of Fixed Effects:
            (Intr) tense1 liftm1
tense1      -0.001
lifetime1    0.000  0.013
tens1:lftm1  0.003  0.002 -0.002
```

4.4 broom.mixed

- the broom.mixed package also contains useful functions for reporting lme4 mixed models
  - broom.mixed::tidy() produces the model estimates for fixed and random effects
  - if you only want fixed effects, use the argument effects = "fixed"
  - if you only want random effects, use the argument effects = "ran_pars"

4.4.1 tidy()

```
tidy_lmer_lifetime <- broom.mixed::tidy(lmer_lifetime)
```

```
tidy_lmer_lifetime
```

```
# A tibble: 7 x 6
  effect   group    term             estimate std.error statistic
  <chr>    <chr>    <chr>               <dbl>     <dbl>     <dbl>
1 fixed    <NA>     (Intercept)        5.63      0.0724     77.7
2 fixed    <NA>     tense1             0.0379    0.0339      1.12
3 fixed    <NA>     lifetime1         -0.103     0.0339     -3.04
4 fixed    <NA>     tense1:lifetime1  -0.0959    0.0678     -1.41
5 ran_pars item_id  sd__(Intercept)    0.0837   NA          NA
6 ran_pars px       sd__(Intercept)    0.185    NA          NA
7 ran_pars Residual sd__Observation    0.394    NA          NA
```

4.5 lmerTest

- is something missing from our model summary?
  - which effects were statistically significant ($p < .05$)?
- calculating p-values is not trivial for mixed models
  - lme4 doesn't do it for linear models
- we can use the lmerTest package, which also has lme4

```
lmerTest_lifetime <-
  lmerTest::lmer(log(fp)  ~ tense * lifetime +
         (1|px) + (1|item_id),
       subset = region == "verb" & fp > 0,
       data = df_lifetime)
```

```
tidy_lmerTest_lifetime <- broom.mixed::tidy(lmerTest_lifetime)
```

```
tidy_lmerTest_lifetime
```

```
# A tibble: 7 x 8
  effect   group    term           estimate std.error statistic     df   p.value
  <chr>    <chr>    <chr>             <dbl>     <dbl>     <dbl>  <dbl>     <dbl>
1 fixed    <NA>     (Intercept)      5.63      0.0724     77.7    6.20  1.65e-10
2 fixed    <NA>     tense1           0.0379    0.0339      1.12 471.    2.63e- 1
3 fixed    <NA>     lifetime1       -0.103     0.0339     -3.04 468.    2.47e- 3
4 fixed    <NA>     tense1:lifeti~  -0.0959    0.0678     -1.41 471.    1.58e- 1
5 ran_pars item_id  sd__(Intercep~   0.0837   NA          NA     NA    NA
6 ran_pars px       sd__(Intercep~   0.185    NA          NA     NA    NA
7 ran_pars Residual sd__Observati~   0.394    NA          NA     NA    NA
```

4.5.1 Publication-ready tables

- we've already seen how to produce publication-ready tables
  - TASK: produce publication-ready tables using `knitr` and `kableExtra` for:
    - a table of the model estimates of the fixed effects of our model
    - a table of the model estimates of the random effects of our model
    - a table of the with both the fixed and random effects
- produce one of these tables as LaTeX code (if you haven't already)
  - can you then produce this table in Overleaf?

4.5.2 Example

```
tidy_lmerTest_lifetime |>
  mutate(p.value = case_when(
    p.value < .001 ~ "< .001",
    TRUE ~ as.character(round(p.value,3))
  )) |>
  knitr::kable(digits = 2) |>
  kableExtra::kable_styling()
```

Table 2: Example model summary table using `broom.mixed::tidy()`, `knitr::kable()`, and `kableExtra::kable_styling()`

effect	group	term	estimate	std.error	statistic	df	p.value
fixed	NA	(Intercept)	5.63	0.07	77.74	6.20	< .001
fixed	NA	tense1	0.04	0.03	1.12	470.73	0.263
fixed	NA	lifetime1	-0.10	0.03	-3.04	468.30	0.002
fixed	NA	tense1:lifetime1	-0.10	0.07	-1.41	470.76	0.158
ran_pars	item_id	sd__(Intercept)	0.08	NA	NA	NA	NA
ran_pars	px	sd__(Intercept)	0.18	NA	NA	NA	NA
ran_pars	Residual	sd__Observation	0.39	NA	NA	NA	NA

4.6 Save our model

- let's store our model locally for future use

````
```{r}
#| eval: false
lmerTest_lifetime <- saveRDS(lmerTest_lifetime, here::here("models", "lmerTest_lifetime"))
```
````

- another tip: you don't want to re-run and re-save your

5 Citing resources

- you should cite the packages you used to fit your models
  - as well as other package versions used (but `SessionInfo()` also takes care of this)
- to cite packages in-text, you can first extract the citation: with

```
citation(package = "lme4")
```

```
To cite lme4 in publications use:

  Douglas Bates, Martin Maechler, Ben Bolker, Steve Walker (2015).
  Fitting Linear Mixed-Effects Models Using lme4. Journal of
  Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

A BibTeX entry for LaTeX users is
```

```
  @Article{,
    title = {Fitting Linear Mixed-Effects Models Using {lme4}},
    author = {Douglas Bates and Martin M{\"a}chler and Ben Bolker and Steve Walker},
    journal = {Journal of Statistical Software},
    year = {2015},
    volume = {67},
    number = {1},
    pages = {1--48},
    doi = {10.18637/jss.v067.i01},
  }
```

- or as BibTeX citation:

```
print(citation(package = "lme4"), bibtex=TRUE)
```

```
To cite lme4 in publications use:

  Douglas Bates, Martin Maechler, Ben Bolker, Steve Walker (2015).
  Fitting Linear Mixed-Effects Models Using lme4. Journal of
  Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

A BibTeX entry for LaTeX users is

  @Article{,
    title = {Fitting Linear Mixed-Effects Models Using {lme4}},
    author = {Douglas Bates and Martin M{\"a}chler and Ben Bolker and Steve Walker},
    journal = {Journal of Statistical Software},
    year = {2015},
    volume = {67},
    number = {1},
    pages = {1--48},
    doi = {10.18637/jss.v067.i01},
  }
```

- then copy it to your `.bib` file, add a citation reference key (e.g., `bates_fitting_2015`), and cite it in-text with `@bates_fitting_2015`, which will be formatted as Bates, Mächler, et al. (2015)
  - alternatively, save the package reference in Zotero and access it using `rbbt`
  - e.g., copy the DOI then find the Zotero button for 'Add Item by Identifier'

5.1 Citing packages

- would usually be done in Data Analysis

```
Linear mixed models were fit using the `lmerTest` package which produces the Satterwaite appr
```

> Linear mixed models were fit using the `lmerTest` package which produces the Satterwaite approximation for degrees of freedom to calculate *p*-values (Kuznetsova et al., 2017).

6 How to report

- you would typically have a (sub)section called Data Analysis which includes all the steps taken prior to inspecting your results
  - i.e., how you prepared and analysed your data
  - this should be information stated in a pre-registration but in the past tense
- this would typically be followed by a Results section
  - containing the results from these cleaning/analysis steps

6.1 Data analysis

- should include, e.g.,
  - model summaries in tables and/or in-text
  - any inclusion/exclusion criteria for participants or observations
  - any transformations performed on dependent or independent variables
    - e.g., we log-transformed first-pass reading times, and used sum contrast coding for our two 2-level predictors
    - we should therefore define which levels were coded as what value (+/-0.5)
- model selection
  - what model did you start with? E.g., maximal model given the data and research questions (à la Barr et al., 2013)
  - how did you handle convergence issues? E.g., parsimonious model selection (à la Bates, Kliegl, et al., 2015)

6.2 Results

- this would be followed by a section called Results, which reports e.g.,
  - the number of participants and trials ex-/included in analyses
  - possibly raw means and 95% confidence intervals
  - model formula
  - information about the number of observations per model
- a structure overview of model results

6.3 Example model report

- in an Rmarkdown script where you write up a paper or thesis, you could include the following hidden code chunk
  - i.e., set `{r, echo = FALSE}`, which is slightly different than the Quarto `#| eval: false` chunk option syntax

```
pacman::p_load(lme4, lmerTest, tidyverse)

lmerTest_lifetime <- readRDS(here::here("models", "lmerTest_lifetime"))
```

```
A linear mixed model was fit to log-transformed first-pass reading times, with `r summary(lme

An effect of *lifetime* was found in first-pass reading times at the verb region (Est = `r s
```

6.4 Output

> A linear mixed model was fit to log-transformed first-pass reading times, with 7 participants and 80 items. The final model formula was `log(fp) ~ tense * lifetime + (1 | px) + (1 | item_id)`.
>
> An effect of *lifetime* was found in first-pass reading times at the verb region (Est = -0.1, t = -3, p = 0.0025). A significant effect of *tense* was not found found (Est = 0.038, t = 1.1, p = 0.26), nor an interaction of *lifetime* and *tense* (Est = -0.096, t = -1.4, p = 0.16).

Learning objectives

Today we...

- run our first linear (mixed) model
- produce model summary tables
- report analyses and cite packages
- report model results reproducibly

Session Info

```
print(sessionInfo(), locale = F)
```

```
R version 4.4.0 (2024-04-24)
Platform: aarch64-apple-darwin20
Running under: macOS Ventura 13.2.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  L

attached base packages:
[1] stats     graphics  grDevices datasets  utils     methods   base

other attached packages:
 [1] lmerTest_3.1-3      brms_2.21.5         Rcpp_1.0.12
 [4] broom.mixed_0.2.9.5 broom_1.0.6         lme4_1.1-35.3
 [7] Matrix_1.7-0        lubridate_1.9.3     forcats_1.0.0
[10] stringr_1.5.1       dplyr_1.1.4         purrr_1.0.2
[13] readr_2.1.5         tidyr_1.3.1         tibble_3.2.1
[16] ggplot2_3.5.1       tidyverse_2.0.0

loaded via a namespace (and not attached):
 [1] tidyselect_1.2.1     viridisLite_0.4.2    loo_2.7.0
 [4] fastmap_1.2.0        tensorA_0.36.2.1     pacman_0.5.1
 [7] digest_0.6.35        timechange_0.3.0     estimability_1.5.1
[10] lifecycle_1.0.4      magrittr_2.0.3       posterior_1.5.0
[13] compiler_4.4.0       rlang_1.1.4          tools_4.4.0
[16] utf8_1.2.4           yaml_2.3.8           knitr_1.47
[19] bridgesampling_1.1-2 bit_4.0.5            here_1.0.1
```

```
[22] xml2_1.3.6           abind_1.4-5          numDeriv_2016.8-1.1
[25] withr_3.0.0          grid_4.4.0           fansi_1.0.6
[28] xtable_1.8-4         colorspace_2.1-0     future_1.33.2
[31] globals_0.16.3       emmeans_1.10.2       scales_1.3.0
[34] MASS_7.3-60.2        cli_3.6.2            mvtnorm_1.2-5
[37] rmarkdown_2.27       crayon_1.5.2         generics_0.1.3
[40] RcppParallel_5.1.7   rstudioapi_0.16.0    tzdb_0.4.0
[43] minqa_1.2.7          splines_4.4.0        bayesplot_1.11.1
[46] parallel_4.4.0       matrixStats_1.3.0    vctrs_0.6.5
[49] boot_1.3-30          jsonlite_1.8.8       hms_1.1.3
[52] bit64_4.0.5          listenv_0.9.1        systemfonts_1.1.0
[55] glue_1.7.0           parallelly_1.37.1    nloptr_2.0.3
[58] codetools_0.2-20     distributional_0.4.0 stringi_1.8.4
[61] gtable_0.3.5         munsell_0.5.1        furrr_0.3.1
[64] pillar_1.9.0         htmltools_0.5.8.1    Brobdingnag_1.2-9
[67] R6_2.5.1             rprojroot_2.0.4      kableExtra_1.4.0
[70] vroom_1.6.5          evaluate_0.23        lattice_0.22-6
[73] backports_1.5.0      renv_1.0.7           rstantools_2.4.0
[76] svglite_2.1.3        coda_0.19-4.1        nlme_3.1-165
[79] checkmate_2.3.1      xfun_0.44            pkgconfig_2.0.3
```

References

Barr, D. J., Levy, R., Scheepers, C., & Tily, H. J. (2013). Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, *68*(3), 255–278. https://doi.org/10.1016/j.jml.2012.11.001

Bates, D., Kliegl, R., Vasishth, S., & Baayen, H. (2015). Parsimonious Mixed Models. *arXiv Preprint*, 1–27. https://doi.org/10.48550/arXiv.1506.04967

Bates, D., Mächler, M., Bolker, B. M., & Walker, S. C. (2015). Fitting linear mixed-effects models using Lme4. *Journal of Statistical Software*, *67*(1). https://doi.org/10.18637/jss.v067.i01

Kuznetsova, A., Brockhoff, P. B., & Christensen, R. H. B. (2017). **lmerTest** Package: Tests in Linear Mixed Effects Models. *Journal of Statistical Software*, *82*(13). https://doi.org/10.18637/jss.v082.i13