live coding (lecture 2)

18 November 2025

```
library(ggplot2); theme_set(theme_bw())
## requires ggplot2 version >= 4.0.0
library(lme4)
library(glmmTMB)
library(lmerTest)
library(ggplot2)
library(tidyverse)
library(plotly)
library(buildmer)
library(broom.mixed)
library(bbmle)
library(DHARMa)
library(effects)
```

You may want to get previously fitted models (this uses the much-maligned `attach()` function)

```
dd <- readRDS("data/ecoreg.rds")
mod_list <- readRDS("outputs/mod_list.rds")
attach(mod_list)
buildmer_list <- readRDS("outputs/buildmer.rds")
attach(buildmer_list)
confint_list <- readRDS("outputs/confint.rds")
attach(confint_list)
confint_tidy <- readRDS("outputs/confint_tidy.rds")
```

(These time-consuming fits etc. are actually done using `lecture2_batch.R`)

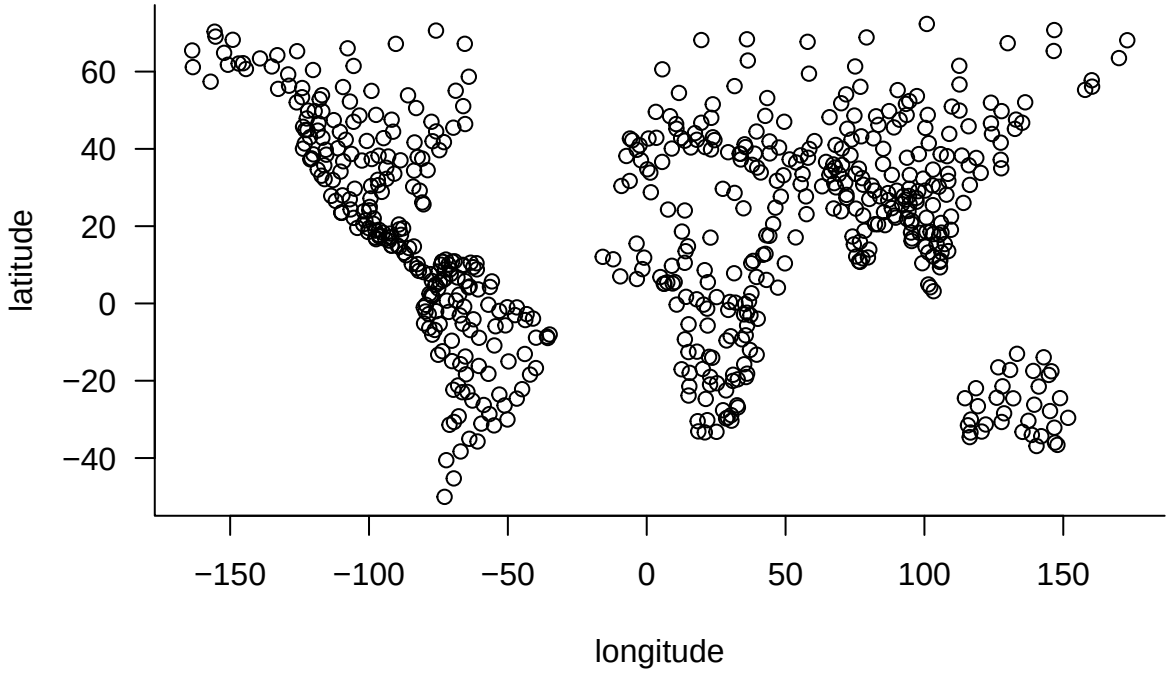
introduction to the example

From Moritz et al. (2023): 620 global observations

Re

Sampling locations/ecoregions:

```
par(las=1, bty = "l")
plot(y ~ x, data = dd, xlab = "longitude", ylab = "latitude")
```



(not bothering to do a proper projection, land boundaries etc.)

exploration

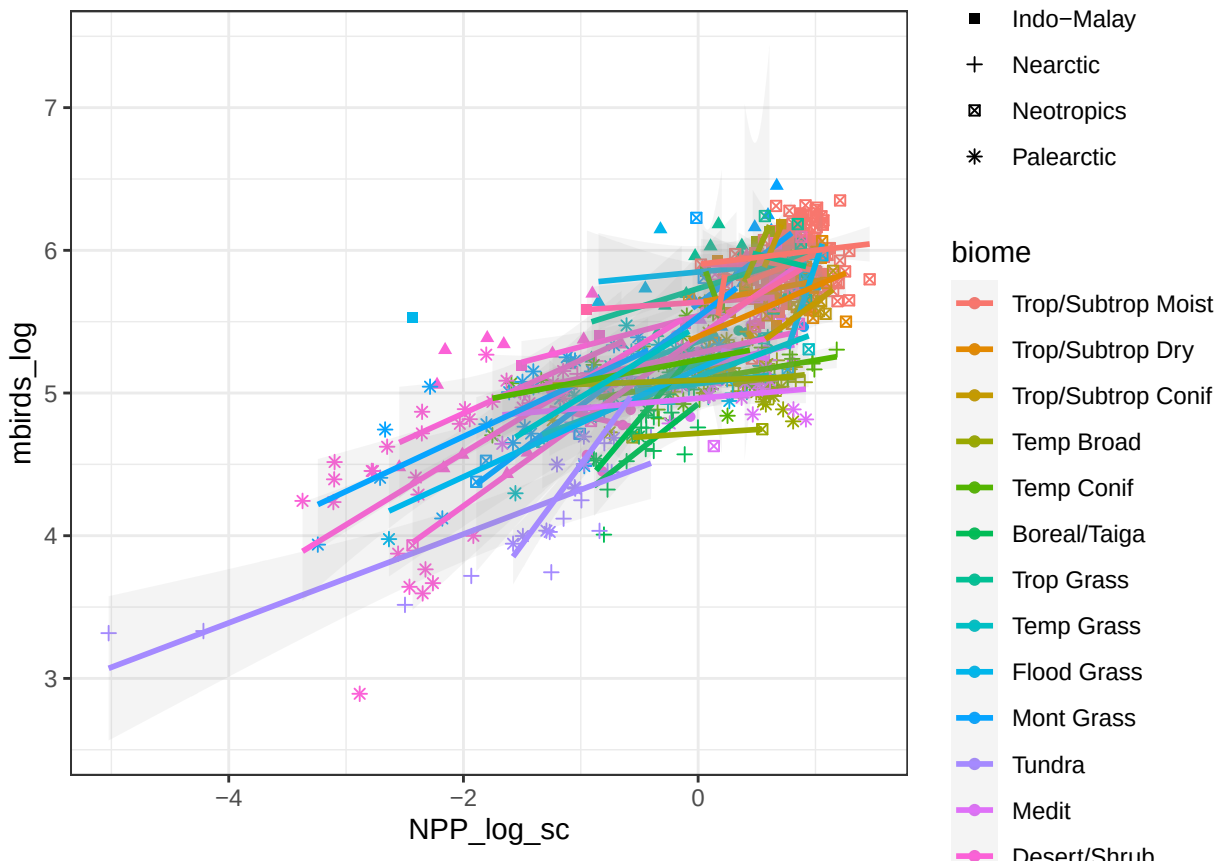
```
dd <- readRDS("data/ecoreg.rds")
```

Draw figures: `ggplot` is particular good for its flexibility — you can easily switch the way you incorporate different predictors (as facets, colours, etc.).

- plot high-dimensional data using all possible mappings - x/y position, colour, shape, line type, plus facets ("small multiples à la Tufte)
- flexibility to switch among different views of the data on the fly
- overlay some kinds of summary statistics/summaries on the fly (means, regression lines etc.)
- tools for grouping (by facet, with `geom_ellipse` or `ggalt::geom_encircle()` ...)

Competitors with similar capabilities include `tinyplot` (R): `plotnine`, `seaborn` (Python); `gadfly`, `AlgebraOfGraphic` (Julia), but `ggplot` is mature (e.g. see ggplot extensions gallery)

```
gg0 <- ggplot(dd, aes(NPP_log_sc, mbirds_log, colour = biome, shape = flor_realms)) +
  geom_point() +
  geom_smooth(method = "lm", aes(group = biome_FR), alpha = 0.1)
print(gg0)
```



Organize plots by biome or floristic realm:

```
gg1 <- gg0 + facet_wrap(~biome)
gg2 <- gg0 + facet_wrap(~flor_realms)
```

With pop-up tooltips (nice for identifying points interactively):

```
ggplotly(gg2)
```

(could also try out `ggiraph` for this?)

model fitting

full model

This is the full model used in Moritz et al. (2023). The fixed effects include all two-way interactions between the covariates of interest (NPP, Fire, annual coefficients of variation of each, all appropriately scaled and/or log-transformed). Random effects include main effects only, varying at the three levels of realms (e.g. "Afrotropics", biomes (e.g. "Tropical grassland"), and their interaction (e.g. "Tropical grasslands in the Afrotropics"). We didn't even try to fit interactions; these covariance matrices are already 5x5 (15 parameters each ...)

I actually used text manipulation to set up the formula, so I could build lots of variations on the model. This makes the code more compact and reproducible (less cutting and pasting), at the expense of transparency.

Here I'm analyzing just the bird data (`mbirds_log`); for the paper we analyzed diversity of birds, mammals and amphibians separately.

```
form <- mbirds_log ~ (NPP_log_sc + Feat_log_sc + NPP_cv_sc + Feat_cv_sc)^2 +
  (1 + NPP_log_sc + Feat_log_sc + NPP_cv_sc + Feat_cv_sc | biome) +
  (1 + NPP_log_sc + Feat_log_sc + NPP_cv_sc + Feat_cv_sc | flor_realms) +
  (1 + NPP_log_sc + Feat_log_sc + NPP_cv_sc + Feat_cv_sc | biome_FR)
```

Fit the model and take a first glance at the RE covariance matrix:

```
m_full <- lmer(form, data = dd)
```

(about 10 seconds)

```
VarCorr(m_full)
```

```
##  Groups      Name        Std.Dev. Corr
##  biome_FR    (Intercept) 0.107644
##              NPP_log_sc  0.122015  0.055
##              Feat_log_sc 0.024896 -0.975 -0.276
##              NPP_cv_sc   0.034689 -0.999 -0.015  0.965
##              Feat_cv_sc  0.061427  0.981  0.247 -1.000 -0.972
##  biome       (Intercept) 0.125732
##              NPP_log_sc  0.158137 -0.470
##              Feat_log_sc 0.013087 -0.996  0.389
##              NPP_cv_sc   0.091421 -0.512  0.999  0.434
##              Feat_cv_sc  0.044459  0.901 -0.806 -0.859 -0.834
##  flor_realms (Intercept) 0.236549
##              NPP_log_sc  0.130949 -0.346
##              Feat_log_sc 0.120903  0.608 -0.955
##              NPP_cv_sc   0.032681  0.012  0.908 -0.764
##              Feat_cv_sc  0.077954  0.357 -0.293  0.360  0.045
##  Residual                0.196065
```

An example of text manipulation, used to illustrate *more* maximal models (4-way interactions in the fixed effects, two-way interactions in the random effects terms)

```
vars <- c("NPP_log_sc", "Feat_log_sc", "NPP_cv_sc", "Feat_cv_sc")
all_vars <- paste(vars, collapse = "+")

max_form <- sprintf("(%s)^4 + ((%s)^2 | biome) + ((%s)^2 | flor_realms) + ((%s)^2 | biome_FR)",
                    all_vars, all_vars, all_vars, all_vars)
max_form2 <- reformulate(max_form, response = "mbirds_log")
```

Fit the ridiculously full model (use `calc.derivs=FALSE` to skip gradient/Hessian checking, speeding things up)

```
m_fullmax <- lmer(max_form2, data = dd, control = lmerControl(calc.derivs = FALSE))
```

(this model takes about 100 minutes to fit ...)

buildmer

Trying out the `buildmer` package to automatically reduce the model:

```
m_buildmer <- buildmer(form, data = dd)
```

Decides on intercepts only at each stage. Unfortunately it also does model selection on the fixed effects by default, which I **strongly** disagree with (for philosophy about model selection/averaging see Bolker (2024))

Use `include` to force `buildmer` to keep all of the fixed-effect terms:

```
m_buildmer_fix <- buildmer(form, data = dd,
                           buildmerControl =
                             buildmerControl(
                               include= ~(NPP_log_sc + Feat_log_sc + NPP_cv_sc + Feat_cv_sc)^2))
```

For birds and amphibians, the selected model included independent effects of the predictors at the biome/realm interaction level and intercept-only (mean diversity) variation at the biome and realm levels; for mammals, the (independent) predictor effects were included at the realm level, with mean- diversity effects at the biome and biome/realm levels.

In Moritz et al. (2023), we fitted all combinations of [intercept-only, diagonal, full] at each level [biome, realm, biome × realm]: 3 choices at each of three hierarchical levels → 81 sub-models to fit. We chose the *minimum-AIC*, *non-singular* model, which turned out to be a diagonal covariance matrix at the (biome×realm) level and intercepts only at the other two levels:

```
form2 <- mbirds_log ~ (NPP_log_sc + Feat_log_sc + NPP_cv_sc + Feat_cv_sc)^2 +
  (1 + NPP_log_sc + Feat_log_sc + NPP_cv_sc + Feat_cv_sc || biome_FR) +
  (1 | biome) +
  (1 | flor_realms)
```

```
m_moritz <- lmer(form2, data = dd)
```

For comparison, let's try this in `glmmTMB` as well:

```
m_glmmTMB <- glmmTMB(form2, data = dd, REML = TRUE)
```

Kliegl method

Check the individual `VarCorr` terms:

```
sapply(VarCorr(m_full), det)
```

```
##      biome_FR         biome   flor_realms
## -2.150556e-41  3.712293e-43 -3.026381e-33
```

All three terms are singular (determinant $\ll 1$), so let's replace them all with diagonal versions:

```
## string-processing a formula is usually a no-no but I'm in a hurry ...
form_diag <- as.formula(gsub("|", "||", deparse1(form), fixed = TRUE))
m_kliegl1 <- lmer(form_diag, data = dd)
```

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

```
## use internal machinery for brevity
lme4:::.prt.VC(VarCorr(m_kliegl1), digits = 3)
```

```
## Random effects:
##  Groups        Name        Std.Dev.
##  biome_FR      Feat_cv_sc  0.0725
##  biome_FR.1    NPP_cv_sc   0.0746
##  biome_FR.2    Feat_log_sc 0.0316
##  biome_FR.3    NPP_log_sc  0.0883
##  biome_FR.4    (Intercept) 0.1285
##  biome         Feat_cv_sc  0.0308
##  biome.1       NPP_cv_sc   0.0000
##  biome.2       Feat_log_sc 0.0000
##  biome.3       NPP_log_sc  0.1321
##  biome.4       (Intercept) 0.1219
```

```
##  flor_realms   Feat_cv_sc  0.0670
##  flor_realms.1 NPP_cv_sc   0.0000
##  flor_realms.2 Feat_log_sc 0.0959
##  flor_realms.3 NPP_log_sc  0.0918
##  flor_realms.4 (Intercept) 0.2201
##  Residual                  0.1955
```

This suggests we should drop `NPP_cv_sc` from the `flor_realms` term; `NPP_cv_sc` and `Feat_log_sc` from `biome`; and keep all terms for `biome_FR`

```
form_kliegl2 <- mbirds_log ~
  (NPP_log_sc + Feat_log_sc + NPP_cv_sc + Feat_cv_sc)^2 +
  (1 + NPP_log_sc + Feat_cv_sc || biome) +
  (1 + NPP_log_sc + Feat_log_sc + Feat_cv_sc || flor_realms) +
  (1 + NPP_log_sc + Feat_log_sc + NPP_cv_sc + Feat_cv_sc || biome_FR)
m_kliegl2 <- lmer(form_kliegl2, data = dd)
isSingular(m_kliegl2)
```

```
## [1] FALSE
```

Can we get away with full covariances once we drop these terms?

```
form_kliegl3 <- as.formula(gsub("||", "|", deparse1(form_kliegl2), fixed = TRUE))
m_kliegl3 <- lmer(form_kliegl3, data = dd)
```

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

```
## Warning: Model failed to converge with 1 negative eigenvalue: -7.1e-01
```

```
sapply(VarCorr(m_kliegl3), det)
```

```
##     biome_FR        biome  flor_realms
## 2.477975e-27 7.076794e-16 0.000000e+00
```

reduced-rank model

```
form_kliegl4 <- mbirds_log ~
  (NPP_log_sc + Feat_log_sc + NPP_cv_sc + Feat_cv_sc)^2 +
  rr(1 + NPP_log_sc + Feat_cv_sc | biome, d=2) +
  rr(1 + NPP_log_sc + Feat_log_sc + Feat_cv_sc | flor_realms, d=3) +
  rr(1 + NPP_log_sc + Feat_log_sc + NPP_cv_sc + Feat_cv_sc | biome_FR, d=4)
m_kliegl4 <- glmmTMB(form_kliegl4, dd, REML=TRUE)
```

```
mod_list <- tibble::lst(m_moritz, m_full, m_fullmax,
                        m_kliegl4,
                        m_buildmer_mod = m_buildmer@model,
                        m_buildmer_fix_mod = m_buildmer_fix@model,
                        m_glmmTMB)
```

```
odir <- "outputs"
if (!dir.exists(odir)) dir.create(odir)
saveRDS(mod_list, file = file.path(odir, "mod_list.rds"))
saveRDS(tibble::lst(m_buildmer, m_buildmer_fix),
        file = file.path(odir, "buildmer.rds"))
```

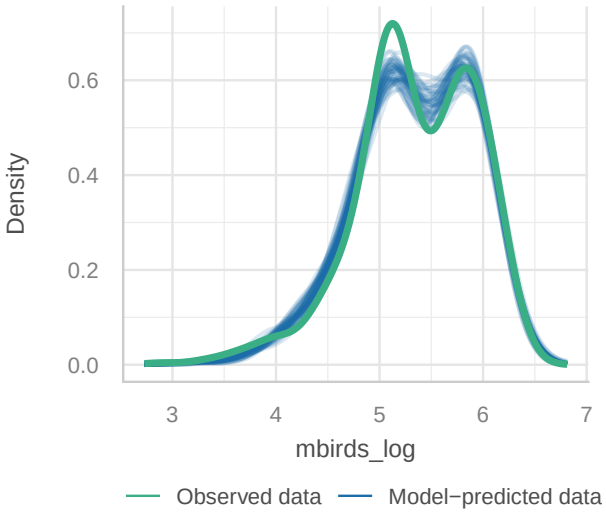
diagnostics

check_model

```
performance::check_model(m_moritz)
```

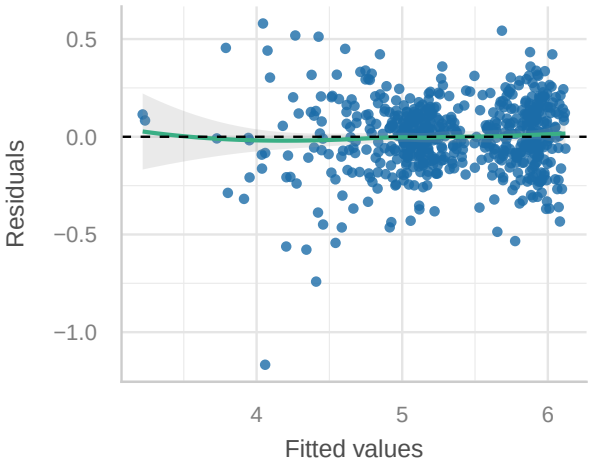
Posterior Predictive Check

Model−predicted lines should resemble observed data line



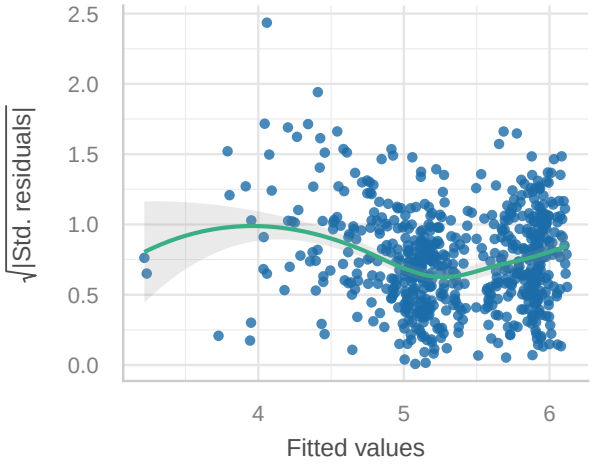
Linearity

Reference line should be flat and horizontal



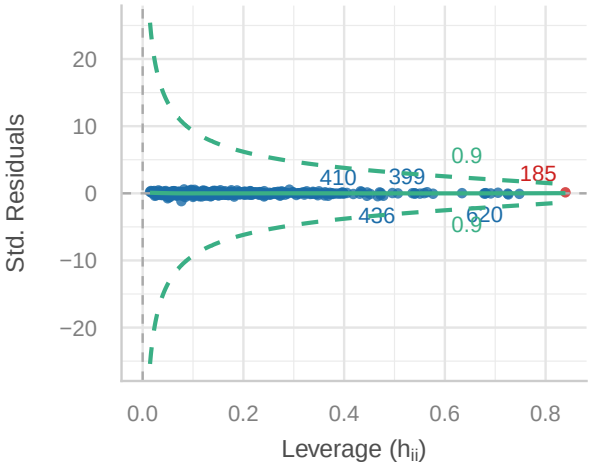
Homogeneity of Variance

Reference line should be flat and horizontal



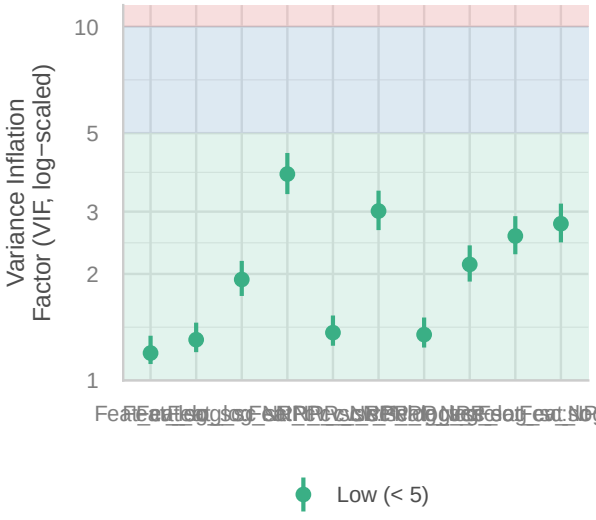
Influential Observations

Points should be inside the contour lines



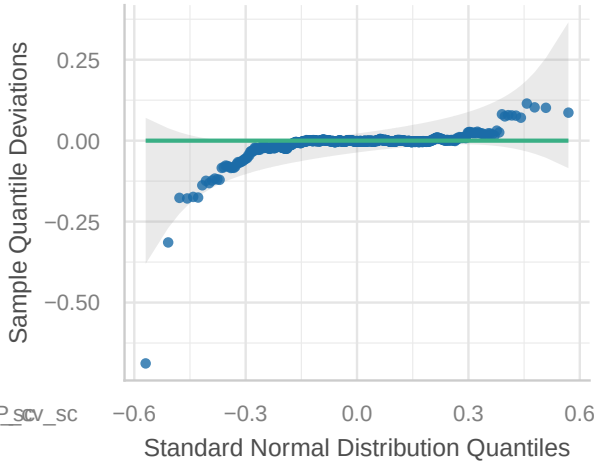
Collinearity

High collinearity (VIF) may inflate parameter uncertainty



Normality of Residuals

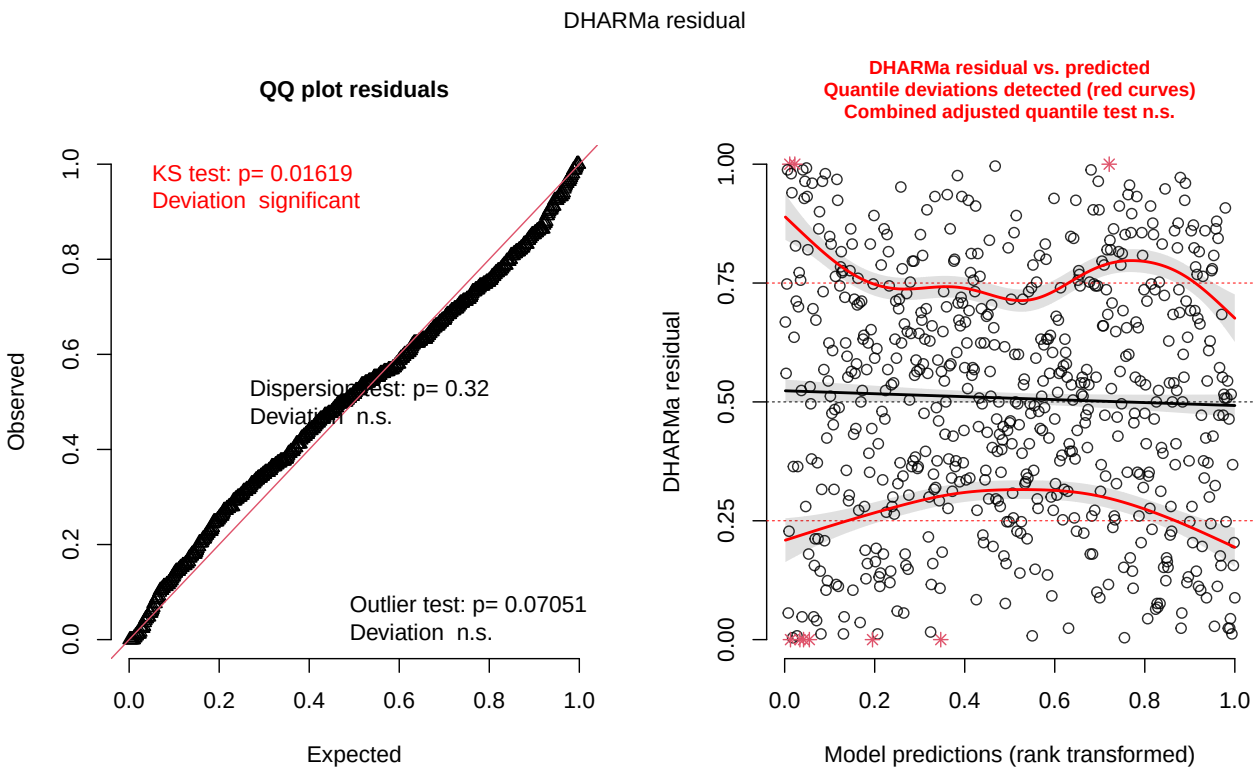
Dots should fall along the line



DHARMa

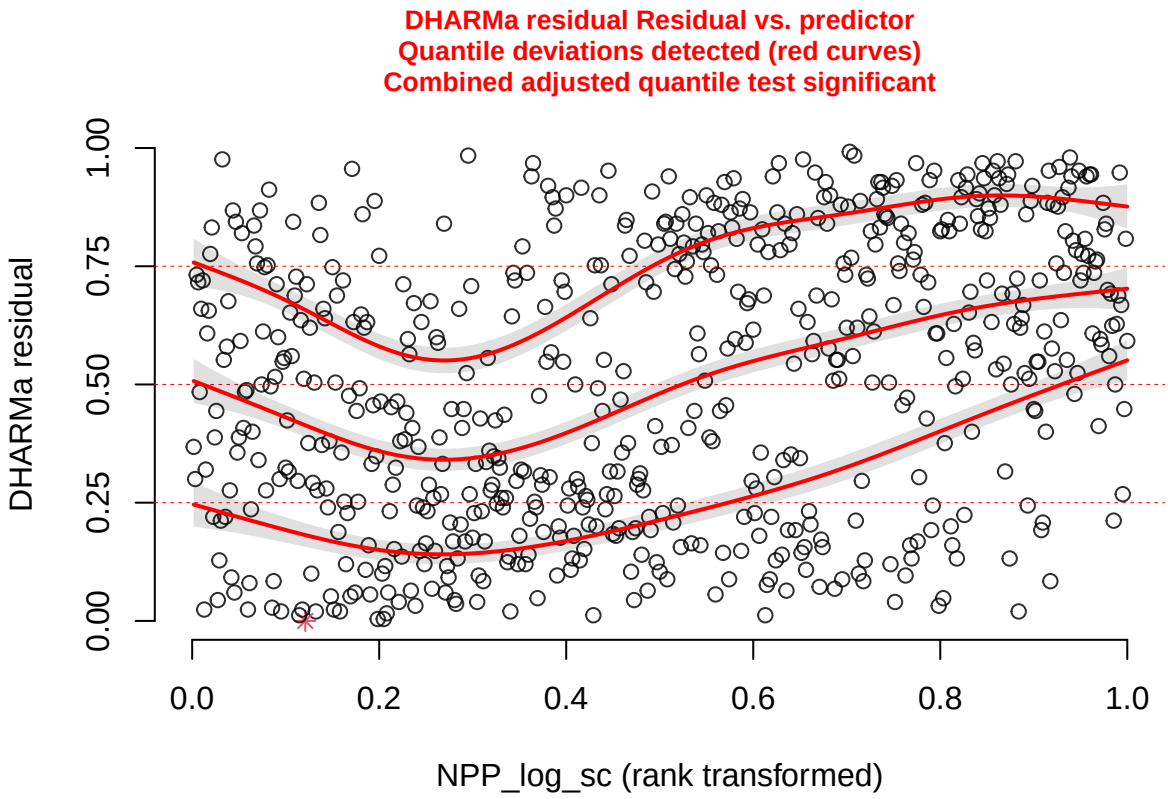
```
pp1 <- DHARMa::simulateResiduals(m_moritz)
## plot(pp1)
pp2 <- DHARMa::simulateResiduals(m_moritz, use.u = TRUE)
plot(pp2)
```

```
## Warning in newton(lsp = lsp, X = G$X, y = G$y, Eb = G$Eb, UrS = G$UrS, L = G$L,
## : Fitting terminated with step failure - check results carefully
```



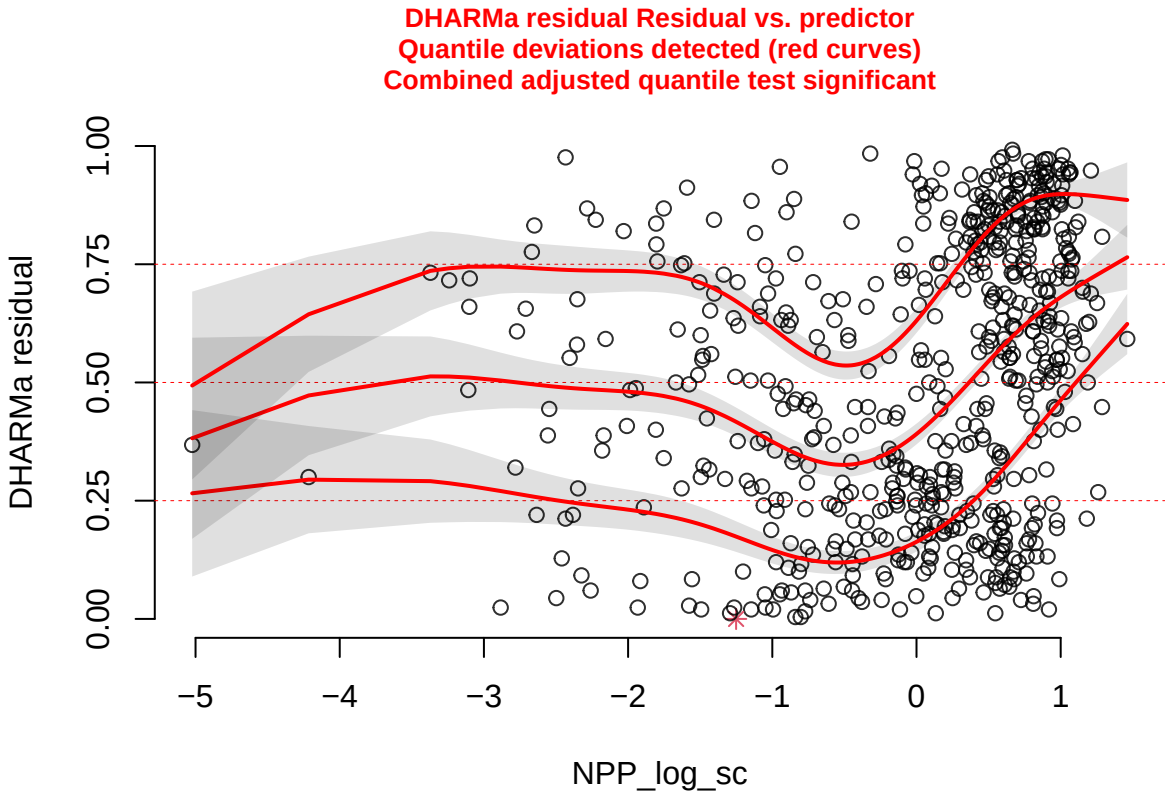
It can be useful to plot against particular covariates, and to turn off the automatic scaling/ranking:

```
plotResiduals(pp1, form = dd$NPP_log_sc)
```




```
plotResiduals(pp1, form = dd$NPP_log_sc, rank = FALSE)
```

```
## Warning in newton(lsp = lsp, X = G$X, y = G$y, Eb = G$Eb, UrS = G$UrS, L = G$L,
## : Fitting terminated with step failure - check results carefully
```



model comparison

```
bbmle::AICtab(mod_list)
```

```
##                     dAIC  df
## m_buildmer_mod       0.0 11
## m_glmmTMB           31.5 19
## m_moritz            31.5 19
## m_kliegl4           43.5 40
## m_buildmer_fix_mod  44.6 16
## m_full              72.0 57
## m_fullmax          334.6 215
```

plotting (coefficient, effects plots)

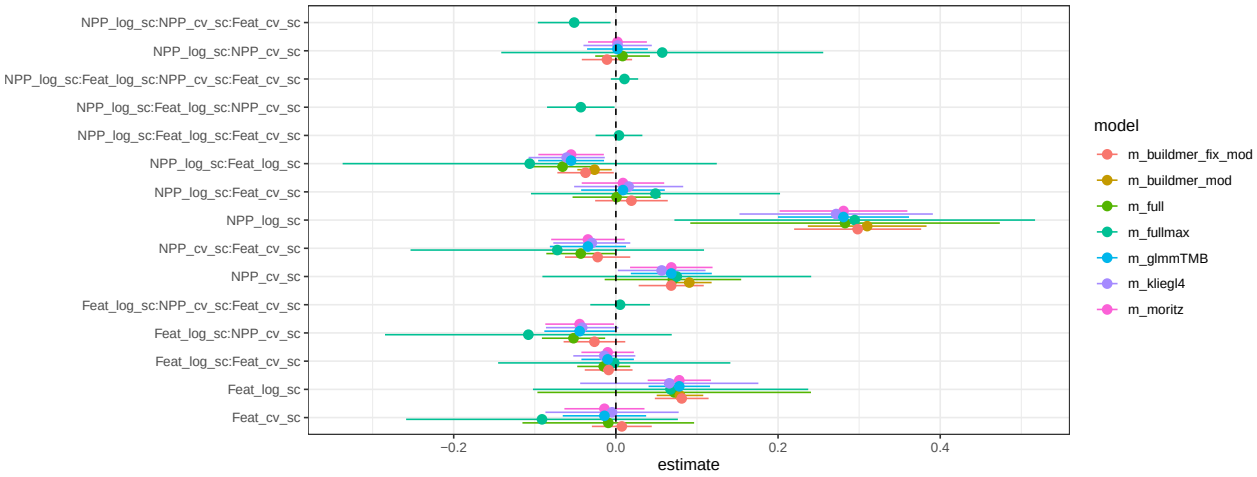
Downstream processing (contrasts, marginal effects, plotting, etc.): `effects`, `broom.mixed`, `sjPlot`, `marginaleffects`, `ggeffects`, `emmeans` (see Mixed Models Task View)

coefficient plots (comparing models)

You **must** (??) decide on a model before comparing parameter estimates — this step is for understanding sensitivity, not for choosing which model to use.

```
tt <-  mod_list |>
  purrr::map_dfr(tidy, effects = "fixed", .id = "model", conf.int = TRUE) |>
  filter(term != "(Intercept)")
```

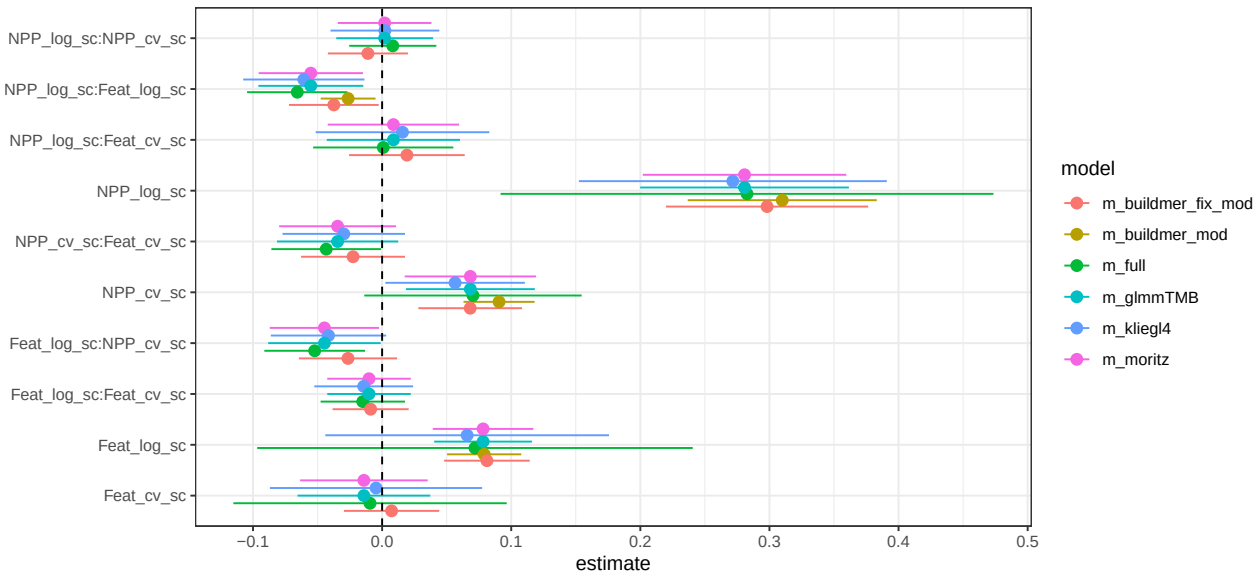
```
gg_coef <- ggplot(tt, aes(estimate, term, colour = model)) +
  geom_pointrange(aes(xmin = conf.low, xmax = conf.high),
                  position = position_dodge(width = 0.75)) +
  geom_vline(xintercept = 0, lty = 2) +
  labs(y = "")
print(gg_coef)
```



Drop "fullmax" model:

```
mod_list2 <- mod_list[names(mod_list) != "m_fullmax"]
tt2 <-  mod_list2 |>
  purrr::map_dfr(tidy, effects = "fixed", .id = "model", conf.int = TRUE) |>
  filter(term != "(Intercept)")

print(gg_coef + tt2)
```



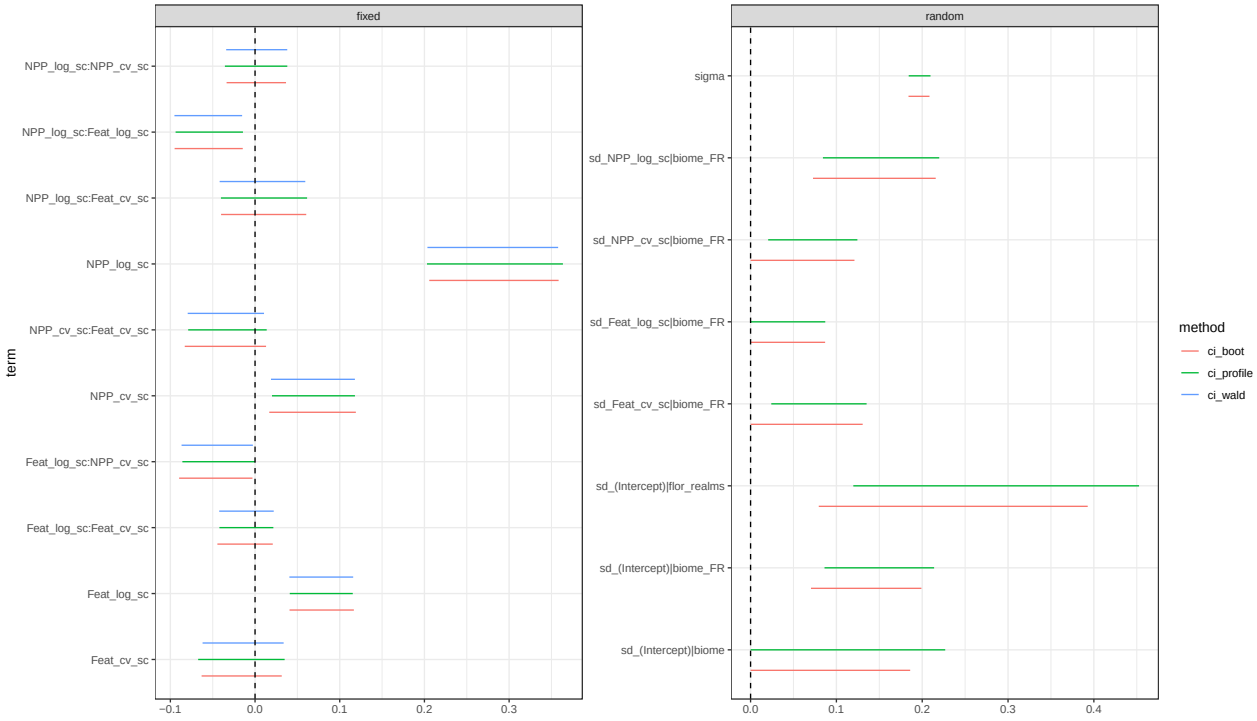
coefficient plots (comparing CI approximations)

```
ci_wald <- confint(m_moritz, method = "Wald", signames = FALSE)
ci_profile <- confint(m_moritz, method = "profile", signames = FALSE)
```

```
ci_boot <- confint(m_moritz, method = "boot", nsim = 1000, parallel = "multicore", ncpus = 6, signames =
confint_list <- tibble::lst(ci_wald, ci_profile, ci_boot)
```

```
confint_tidy <- confint_list |>
  map(as.data.frame) |>
  map(~rownames_to_column(., var = "term")) |>
  bind_rows(.id = "method") |>
  mutate(effect = ifelse(grepl("^(sd|cor|sigma)", term), "random", "fixed")) |>
  rename(lwr = "2.5 %", upr = "97.5 %") |>
  filter(term != "(Intercept)")
```

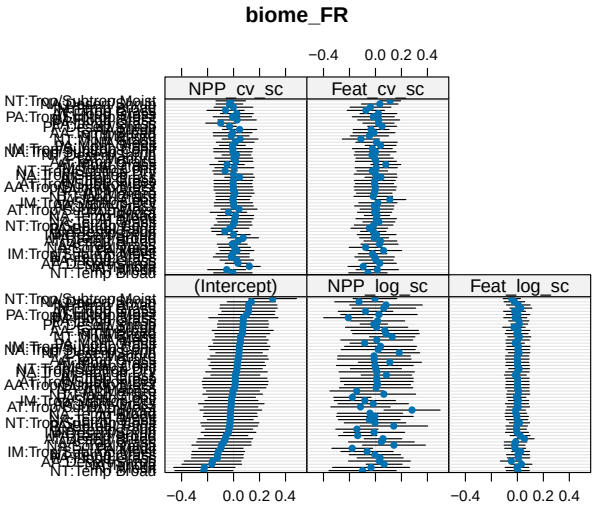
```
ggplot(confint_tidy, aes(xmin = lwr, xmax = upr, y = term, colour = method)) +
  geom_linerange(position = position_dodge(width = 0.75)) +
  geom_vline(lty = 2, xintercept = 0) +
  facet_wrap(~effect, scale = "free")
```

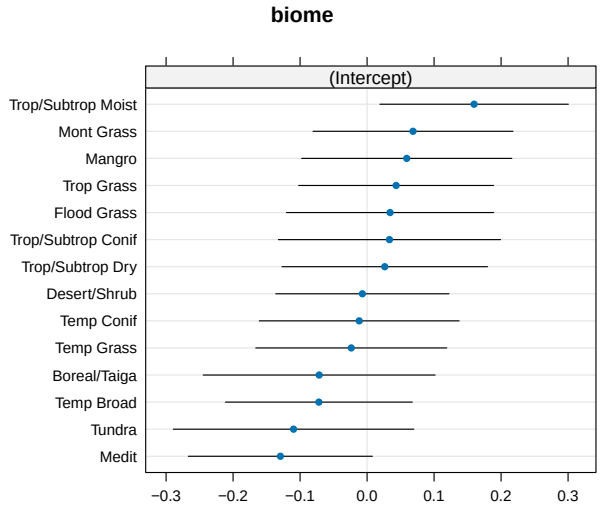


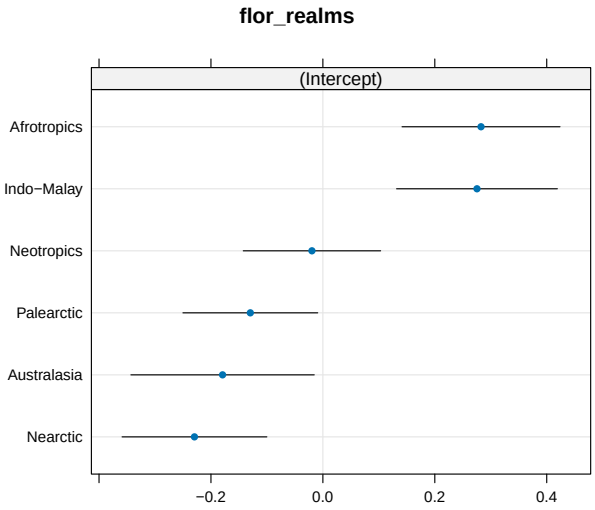
Plotting random effects

'caterpillar plots'

```
cowplot::plot_grid(plotlist = lattice::dotplot(ranef(m_moritz)))
```





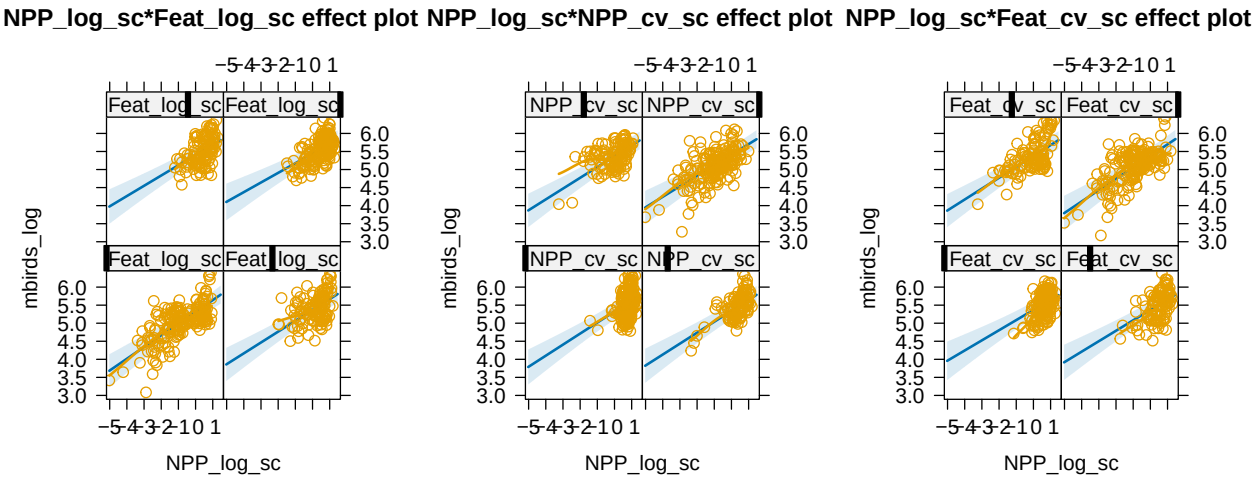


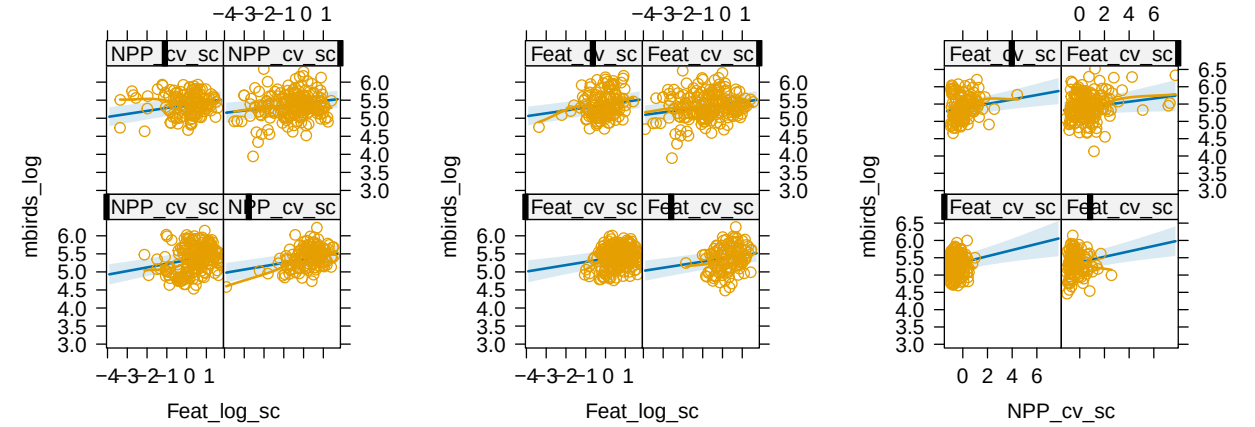
Effects plots

Effects plots, with *partial residuals* (also see the `remef` package, on GitHub).

For conditional and marginal predictions/effects plots, see also `emmeans`/`sjPlot`/`ggeffects`/`marginaleffects` packages ...

```
plot(allEffects(m_buildmer_fix_mod, partial.residuals = TRUE))
```





Would like to try `marginaleffects` but struggling ...

References

Bolker, B. M. (2024). Multimodel approaches are not the best way to understand multifactorial systems. *Entropy*, *26*(6), 506. https://doi.org/10.3390/e26060506

Moritz, M. A., Batllori, E., & Bolker, B. M. (2023). The role of fire in terrestrial vertebrate richness patterns. *Ecology Letters*, *26*(4), 563–574. https://doi.org/10.1111/ele.14177