**New Features of the parameters and see Package**

We’re excited to announce some news from the [easystats-project](https://github.com/easystats/easystats). Two packages were updated recently, the [**parameters**-package](https://easystats.github.io/parameters) and our visualization-toolbox, the [**see**-package](http://easystats.github.io/see).

Before we start introducing some of the new features, we’d like to explain why you need the **see**-package to create plots for functions from other *easystats* packages. So, the **see**-package not only includes additional geoms, color scales and themes for **ggplot2**, but – maybe more important – also provides plot()-methods for many functions from the various *easystats* packages. By separating the plotting functionality from our core packages, packages from the easystats-project don’t rely nor import any other packages! This means that you can safely use them as dependencies in your own packages, without the risk of butterfly effects (a small change in a distant downstream dependency with unexpected upstream consequences).

**Print and Plot your Model Parameters**

The [model\_parameters()](https://easystats.github.io/parameters/reference/model_parameters.html) function in the **parameters**-package is simlar to broom::tidy() – it returns a summary of the model parameters as a clean, consistent data frame. [standardize\_names()](https://easystats.github.io/parameters/reference/standardize_names.html) can be used to return a data frame with column names as they are used in other packages (like **broom**).

model\_parameters() supports many different models, including mixed or Bayesian regression models. It comes with nice print() and plot() methods.

**Examples – Zero Inflated Mixed Models with glmmTMB**

The first example is a zero-inflated mixed model, fitted with the **glmmTMB**-package. model\_parameters() creates separate tables for different model components (like zero-inflated components).

library(glmmTMB)

data(Salamanders)

model <- glmmTMB(

count ~ spp + mined + (1 | site),

ziformula = ~mined,

family = poisson(),

data = Salamanders

)

model\_parameters(model)

## # Fixed Effects component

##

## Parameter | Coefficient | SE | 95% CI | z | df | p

## ------------------------------------------------------------------------

## (Intercept) | -0.36 | 0.28 | [-0.90, 0.18] | -1.30 | 633 | 0.194

## spp [PR] | -1.27 | 0.24 | [-1.74, -0.80] | -5.27 | 633 | < .001

## spp [DM] | 0.27 | 0.14 | [ 0.00, 0.54] | 1.95 | 633 | 0.051

## spp [EC-A] | -0.57 | 0.21 | [-0.97, -0.16] | -2.75 | 633 | 0.006

## spp [EC-L] | 0.67 | 0.13 | [ 0.41, 0.92] | 5.20 | 633 | < .001

## spp [DES-L] | 0.63 | 0.13 | [ 0.38, 0.87] | 4.96 | 633 | < .001

## spp [DF] | 0.12 | 0.15 | [-0.17, 0.40] | 0.78 | 633 | 0.435

## mined [no] | 1.27 | 0.27 | [ 0.74, 1.80] | 4.72 | 633 | < .001

##

## # Zero-Inflated component

##

## Parameter | Coefficient | SE | 95% CI | z | df | p

## ------------------------------------------------------------------------

## (Intercept) | 0.79 | 0.27 | [ 0.26, 1.32] | 2.90 | 633 | 0.004

## mined [no] | -1.84 | 0.31 | [-2.46, -1.23] | -5.87 | 633 | < .001

If only a specific part of the model should be shown, use the component-argument.

model\_parameters(model, component = "zero\_inflated")

## Parameter | Coefficient | SE | 95% CI | z | df | p

## ------------------------------------------------------------------------

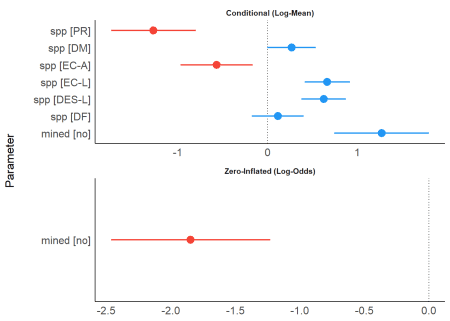
## (Intercept) | 0.79 | 0.27 | [ 0.26, 1.32] | 2.90 | 633 | 0.004

## mined [no] | -1.84 | 0.31 | [-2.46, -1.23] | -5.87 | 633 | < .001

plot() creates a so called “forest plot”. In case of models with multiple components, parameters are separated into facets by model component.

result <- model\_parameters(model)

plot(result)



**Examples – Bayesian Mixed Models with brms**

For the next example, we download a pre-compiled **brms** model to save computation time. For Bayesian models, by default, only “fixed” effects are shown. Using effects = "all" and component = "all" allows us to display random effects and the parameters of the zero-inflated model part as well.

# We download the model to save computation time. Here is the code

# to refit the exact model used below...

# zinb <- read.csv("<http://stats.idre.ucla.edu/stat/data/fish.csv>")

# set.seed(123)

# model <- brm(bf(

# count ~ persons + child + camper + (1 | persons),

# zi ~ child + camper + (1 | persons)

# ),

# data = zinb,

# family = zero\_inflated\_poisson()

# )

brms\_model <- insight::download\_model("brms\_zi\_2")

result <- model\_parameters(brms\_model, effects = "all", component = "all")

result

## # Fixed Effects (Count Model)

##

## Parameter | Median | 89% CI | pd | % in ROPE | ESS | Rhat

## -------------------------------------------------------------------------

## (Intercept) | -0.84 | [-1.44, -0.29] | 96.43% | 2.77% | 562 | 1.009

## persons | 0.84 | [ 0.66, 1.06] | 100% | 0% | 382 | 1.010

## child | -1.15 | [-1.29, -0.98] | 100% | 0% | 1089 | 1.002

## camper | 0.73 | [ 0.58, 0.89] | 100% | 0% | 2724 | 1.000

##

## # Fixed Effects (Zero-Inflated Model)

##

## Parameter | Median | 89% CI | pd | % in ROPE | ESS | Rhat

## -------------------------------------------------------------------------

## (Intercept) | -0.64 | [-1.93, 0.52] | 83.15% | 6.95% | 845 | 1.001

## child | 1.88 | [ 1.40, 2.43] | 100% | 0% | 2322 | 1.001

## camper | -0.83 | [-1.41, -0.24] | 98.95% | 1.70% | 2277 | 1.001

##

## # Random Effects (Count Model)

##

## Parameter | Median | 89% CI | pd | % in ROPE | ESS | Rhat

## ---------------------------------------------------------------------

## persons.1 | -0.01 | [-0.38, 0.28] | 55.33% | 60.50% | 572 | 1.009

## persons.2 | 0.02 | [-0.17, 0.30] | 61.88% | 65.62% | 691 | 1.008

## persons.3 | -0.02 | [-0.26, 0.18] | 61.27% | 67.90% | 340 | 1.011

## persons.4 | 0.00 | [-0.32, 0.33] | 51.38% | 62.12% | 287 | 1.011

##

## # Random Effects (Zero-Inflated Model)

##

## Parameter | Median | 89% CI | pd | % in ROPE | ESS | Rhat

## ----------------------------------------------------------------------

## persons.1 | 1.28 | [ 0.08, 2.70] | 95.73% | 2.15% | 811 | 1.001

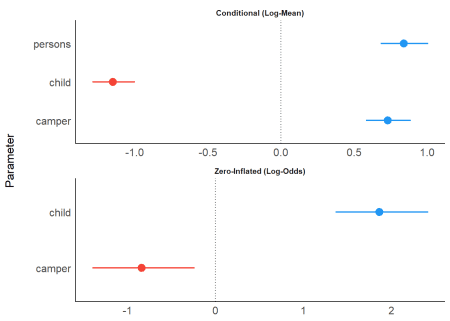
## persons.2 | 0.25 | [-0.90, 1.57] | 66.45% | 12.72% | 759 | 1.001

## persons.3 | -0.18 | [-1.51, 1.01] | 59.67% | 11.28% | 871 | 1.001

## persons.4 | -1.29 | [-2.62, -0.01] | 94.85% | 1.85% | 912 | 1.000

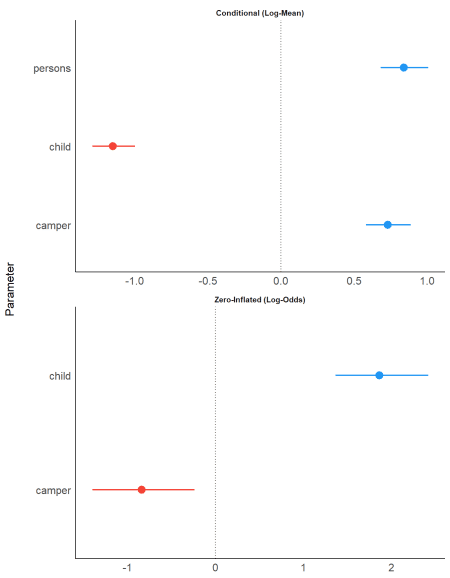
plot() now creates four facets by default. Note that plots from the **parameters**-package create forest plots. If you prefer plots that show the complete posterior distribution, you may rather use functions from the **bayestestRR**-package, as shown [here](https://easystats.github.io/see/articles/bayestestR.html).

plot(result)



You can modify the layout with n\_columns.

plot(result, n\_columns = 1)



**Examples – Meta-Analysis with metafor**

Even results from meta-analyses, using the **metafor**-package, can be visualized with **parameters** and **see**.

library(metafor)

d <- data.frame(

effectsize = c(-0.393, 0.675, 0.282, -1.398),

standarderror = c(0.317, 0.317, 0.13, 0.36)

)

ma <- rma(yi = effectsize, sei = standarderror, method = "REML", data = d)

result <- model\_parameters(ma)

result

## Parameter | Coefficient | SE | 95% CI | z | p | Weight

## ---------------------------------------------------------------------------

## Study 1 | -0.39 | 0.32 | [-1.01, 0.23] | -1.24 | 0.215 | 9.95

## Study 2 | 0.68 | 0.32 | [ 0.05, 1.30] | 2.13 | 0.033 | 9.95

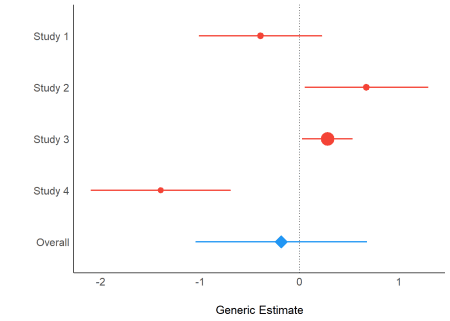
## Study 3 | 0.28 | 0.13 | [ 0.03, 0.54] | 2.17 | 0.030 | 59.17

## Study 4 | -1.40 | 0.36 | [-2.10, -0.69] | -3.88 | < .001 | 7.72

## (Intercept) | -0.18 | 0.44 | [-1.05, 0.68] | -0.42 | 0.676 |

Above you see the coefficients, their standard errors and the “weight”, based on the inverse variance. When you plot the results, the dot-geoms have different sizes, depending on the weight of the study (similar to metafor::forest()).

plot(result)



**Check out** more examples and documentation