Estimating Distributional Models with brms

Paul Bürkner

2022-09-19

Introduction

This vignette provides an introduction on how to fit distributional regression models with **brms**. We use the term *distributional model* to refer to a model, in which we can specify predictor terms for all parameters of the assumed response distribution. In the vast majority of regression model implementations, only the location parameter (usually the mean) of the response distribution depends on the predictors and corresponding regression parameters. Other parameters (e.g., scale or shape parameters) are estimated as auxiliary parameters assuming them to be constant across observations. This assumption is so common that most researchers applying regression models are often (in my experience) not aware of the possibility of relaxing it. This is understandable insofar as relaxing this assumption drastically increase model complexity and thus makes models hard to fit. Fortunately, **brms** uses **Stan** on the backend, which is an incredibly flexible and powerful tool for estimating Bayesian models so that model complexity is much less of an issue.

Suppose we have a normally distributed response variable. Then, in basic linear regression, we specify a predictor term η_μ for the mean parameter μ of the normal distribution. The second parameter of the normal distribution – the residual standard deviation σ – is assumed to be constant across observations. We estimate σ but do not try to *predict* it. In a distributional model, however, we do exactly this by specifying a predictor term η_σ for σ in addition to the predictor term η_μ. Ignoring group-level effects for the moment, the linear predictor of a parameter θ for observation n has the form

$$\eta_{\theta n} = \sum_{i=1}^{K_\theta} b_{\theta i} x_{\theta i n}$$

where $x_{\theta i n}$ denotes the value of the ith predictor of parameter θ for observation n and $b_{\theta i}$ is the ith regression coefficient of parameter θ. A distributional normal model with response variable y can then be written as

$$y_n \sim \mathcal{N}\left(\eta_{\mu n}, \exp(\eta_{\sigma n})\right)$$

We used the exponential function around η_σ to reflect that σ constitutes a standard deviation and thus only takes on positive values, while a linear predictor can be any real number.

A simple distributional model

Unequal variance models are possibly the most simple, but nevertheless very important application of distributional models. Suppose we have two groups of patients: One group receives a treatment (e.g., an antidepressive drug) and another group receives placebo. Since the treatment may not work equally well for all patients, the symptom variance of the treatment group may be larger than the symptom variance of the placebo group after some weeks of treatment. For simplicity, assume that we only investigate the post-treatment values.

```
group <- rep(c("treat", "placebo"), each = 30)
symptom_post <- c(rnorm(30, mean = 1, sd = 2), rnorm(30, mean = 0, sd = 1))
dat1 <- data.frame(group, symptom_post)
head(dat1)
```

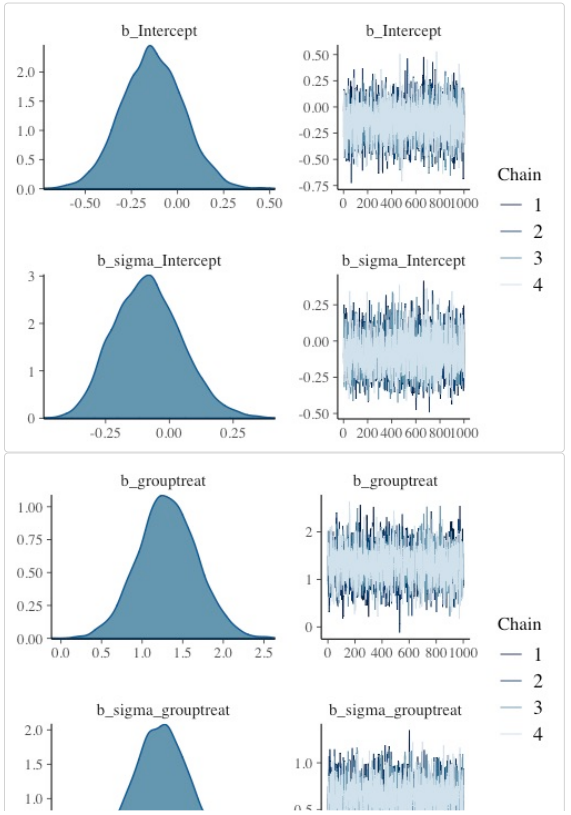
```
  group symptom_post
1 treat    0.7247174
2 treat   -0.2835098
3 treat    1.4097922
4 treat    2.5076797
5 treat    1.3331313
6 treat    3.7490589
```

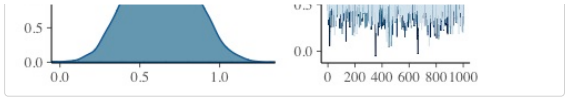
The following model estimates the effect of `group` on both the mean and the residual standard deviation of the normal response distribution.

```
fit1 <- brm(bf(symptom_post ~ group, sigma ~ group),
            data = dat1, family = gaussian())
```

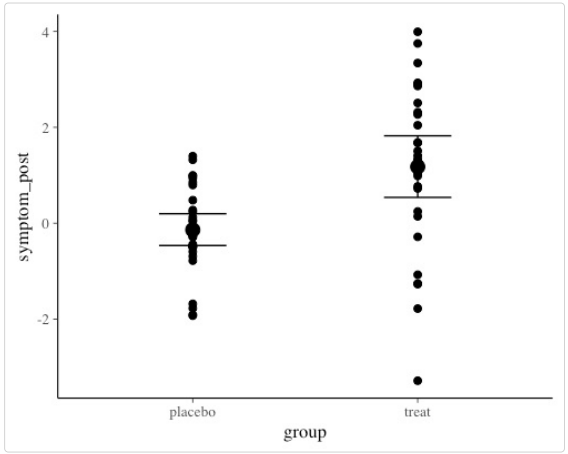
Useful summary statistics and plots can be obtained via

```
summary(fit1)
plot(fit1, N = 2, ask = FALSE)
```





```
plot(conditional_effects(fit1), points = TRUE)
```



The population-level effect `sigma_grouptreat`, which is the contrast of the two residual standard deviations on the log-scale, reveals that the variances of both groups are indeed different. This impression is confirmed when looking at the `conditional_effects` of `group`. Going one step further, we can compute the residual standard deviations on the original scale using the `hypothesis` method.

```
hyp <- c("exp(sigma_Intercept) = 0",
         "exp(sigma_Intercept + sigma_grouptreat) = 0")
hypothesis(fit1, hyp)
```

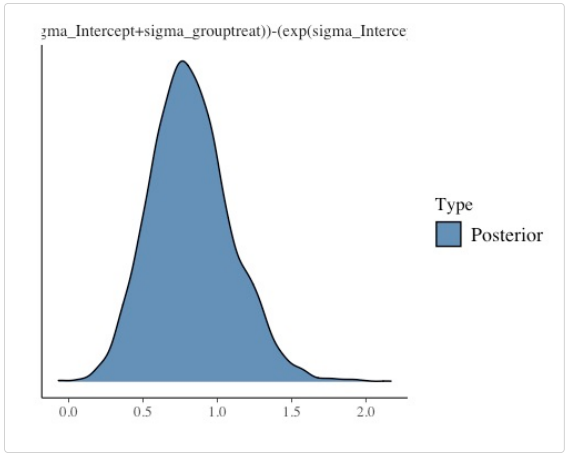
```
Hypothesis Tests for class b:
               Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob Star
1 (exp(sigma_Interc... = 0     0.93      0.13     0.73     1.21         NA        NA    *
2 (exp(sigma_Interc... = 0     1.75      0.25     1.34     2.31         NA        NA    *
---
'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.
'*': For one-sided hypotheses, the posterior probability exceeds 95%;
for two-sided hypotheses, the value tested against lies outside the 95%-CI.
Posterior probabilities of point hypotheses assume equal prior probabilities.
```

We may also directly compare them and plot the posterior distribution of their difference.

```
hyp <- "exp(sigma_Intercept + sigma_grouptreat) > exp(sigma_Intercept)"
(hyp <- hypothesis(fit1, hyp))
```

```
Hypothesis Tests for class b:
               Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob Star
1 (exp(sigma_Interc... > 0     0.82      0.28      0.4      1.3    1332.33         1    *
---
'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.
'*': For one-sided hypotheses, the posterior probability exceeds 95%;
for two-sided hypotheses, the value tested against lies outside the 95%-CI.
Posterior probabilities of point hypotheses assume equal prior probabilities.
```

```
plot(hyp, chars = NULL)
```



Indeed, the residual standard deviation of the treatment group seems to larger than that of the placebo group. Moreover the magnitude of this difference is pretty similar to what we expected due to the values we put into the data simulations.

Zero-Inflated Models

Another important application of the distributional regression framework are so called zero-inflated models. These models are helpful whenever there are more zeros in the response variable than one would naturally expect. For example, if one seeks to predict the number of cigarettes people smoke per day and also includes non-smokers, there will be a huge amount of zeros which, when not modeled appropriately, can seriously distort parameter estimates. Here, we consider an example dealing with the number of fish caught by various groups of people. On the UCLA website (), the data are described as follows: “The state wildlife biologists want to model how many fish are being caught by fishermen at a state park. Visitors are asked how long they stayed, how many people were in the group, were there children in the group and how many fish were caught. Some visitors do not fish, but there is no data on whether a person fished or not. Some visitors who did fish did not catch any fish so there are excess zeros in the data because of the people that did not fish.”

```
zinb <- read.csv("https://paul-buerkner.github.io/data/fish.csv")
head(zinb)
```

```
  nofish livebait camper persons child         xb          zg count
1      1        0      0       1     0 -0.8963146  3.0504048     0
2      0        1      1       1     0 -0.5583450  1.7461489     0
3      0        1      0       1     0 -0.4017310  0.2799389     0
4      0        1      1       2     1 -0.9562981 -0.6015257     0
5      0        1      0       1     0  0.4368910  0.5277091     1
6      0        1      1       4     2  1.3944855 -0.7075348     0
```

As predictors we choose the number of people per group, the number of children, as well as whether the group consists of campers. Many groups may not even try catching any fish at all (thus leading to many zero responses) and so we fit a zero-inflated Poisson model to the data. For now, we assume a constant

zero-inflation probability across observations.

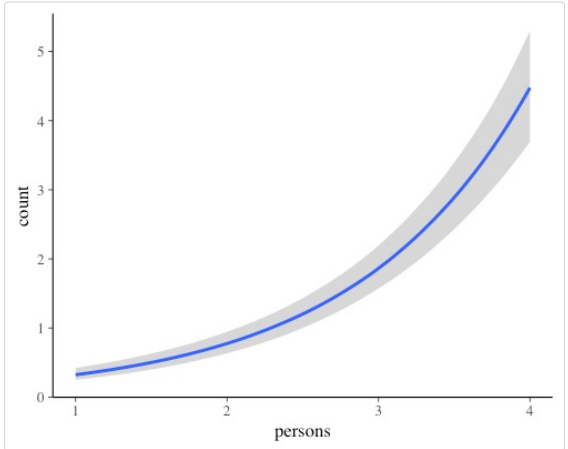
```
fit_zinb1 <- brm(count ~ persons + child + camper,
                 data = zinb, family = zero_inflated_poisson())
```

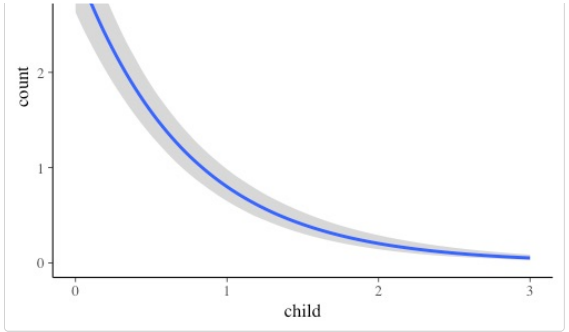
Again, we summarize the results using the usual methods.

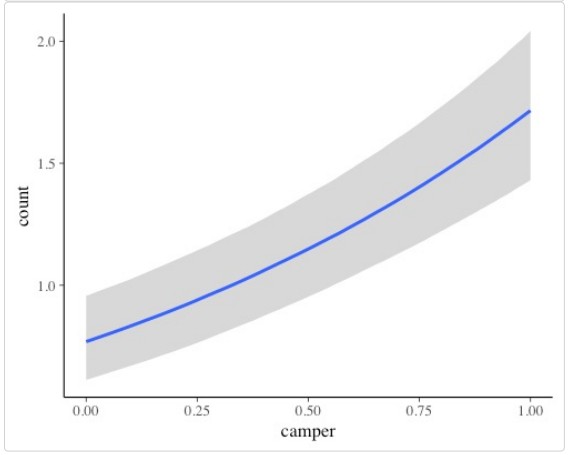
```
summary(fit_zinb1)
```

```
 Family: zero_inflated_poisson
  Links: mu = log; zi = identity
Formula: count ~ persons + child + camper
   Data: zinb (Number of observations: 250)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept    -1.01      0.18    -1.38    -0.66 1.00     2698     2612
persons       0.87      0.05     0.78     0.97 1.00     2839     2603
child        -1.37      0.09    -1.55    -1.19 1.00     2543     2428
camper        0.80      0.09     0.62     0.99 1.00     3266     2670

Family Specific Parameters:
   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
zi     0.41      0.05     0.32     0.50 1.00     2893     2596

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
plot(conditional_effects(fit_zinb1), ask = FALSE)
```







According to the parameter estimates, larger groups catch more fish, campers catch more fish than non-campers, and groups with more children catch less fish. The zero-inflation probability `zi` is pretty large with a mean of 41%. Please note that the probability of catching no fish is actually higher than 41%, but parts of this probability are already modeled by the Poisson distribution itself (hence the name zero-*inflation*). If you want to treat all zeros as originating from a separate process, you can use hurdle models instead (not shown here).

Now, we try to additionally predict the zero-inflation probability by the number of children. The underlying reasoning is that we expect groups with more children to not even try catching fish. Most children are just terribly bad at waiting for hours until something happens. From a purely statistical perspective, zero-inflated (and hurdle) distributions are a mixture of two processes and predicting both parts of the model is natural and often very reasonable to make full use of the data.

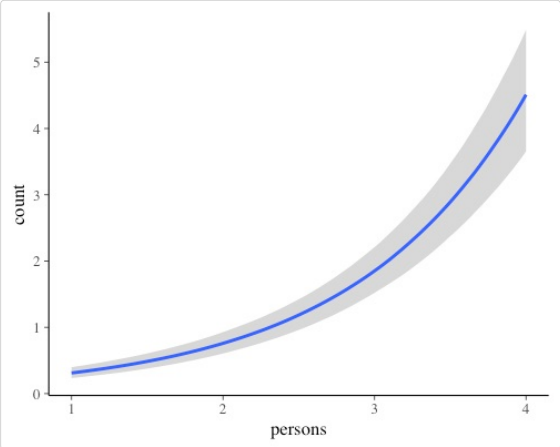
```
fit_zinb2 <- brm(bf(count ~ persons + child + camper, zi ~ child),
                 data = zinb, family = zero_inflated_poisson())
```

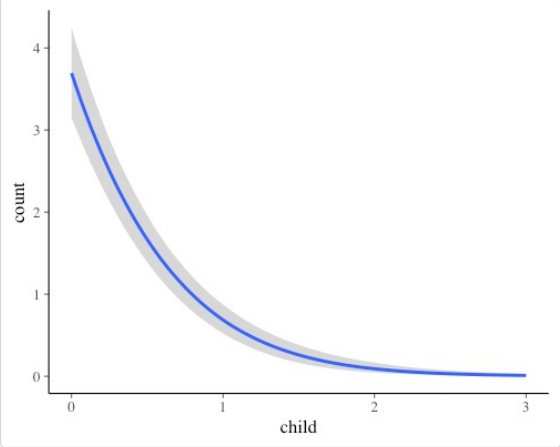
```
summary(fit_zinb2)
```

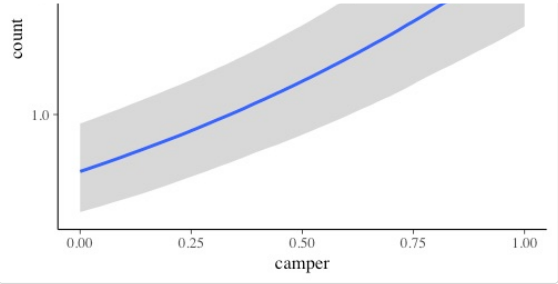
```
 Family: zero_inflated_poisson
  Links: mu = log; zi = logit
Formula: count ~ persons + child + camper
         zi ~ child
   Data: zinb (Number of observations: 250)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000
```

```
Population-Level Effects:
             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept       -1.08      0.18    -1.44    -0.73 1.00     2913     3016
zi_Intercept    -0.96      0.26    -1.50    -0.48 1.00     3813     2853
persons          0.89      0.05     0.81     0.99 1.00     2961     2773
child           -1.18      0.09    -1.36    -1.00 1.00     2565     2677
camper           0.78      0.09     0.60     0.97 1.00     3743     2696
zi_child         1.22      0.27     0.70     1.76 1.00     4043     3060

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
plot(conditional_effects(fit_zinb2), ask = FALSE)
```







To transform the linear predictor of `zi` into a probability, **brms** applies the logit-link:

$$logit(zi) = \log\left(\frac{zi}{1 - zi}\right) = \eta_{zi}$$

The logit-link takes values within $[0, 1]$ and returns values on the real line. Thus, it allows the transition between probabilities and linear predictors.

According to the model, trying to fish with children not only decreases the overall number fish caught (as implied by the Poisson part of the model) but also drastically increases your change of catching no fish at all (as implied by the zero-inflation part) most likely because groups with more children are not even trying.

Additive Distributional Models

In the examples so far, we did not have multilevel data and thus did not fully use the capabilities of the distributional regression framework of **brms**. In the example presented below, we will not only show how to deal with multilevel data in distributional models, but also how to incorporate smooth terms (i.e., splines) into the model. In many applications, we have no or only a very vague idea how the relationship between a predictor and the response looks like. A very flexible approach to tackle this problems is to use splines and let them figure out the form of the relationship. For illustration purposes, we simulate some data with the **mgcv** package, which is also used in **brms** to prepare smooth terms.

```
dat_smooth <- mgcv::gamSim(eg = 6, n = 200, scale = 2, verbose = FALSE)
```

```
Gu & Wahba 4 term additive model
```

```
head(dat_smooth[, 1:6])
```

```
          y        x0         x1         x2         x3         f
1  8.515469 0.1905577 0.08812703 0.74807568 0.42217250  7.423304
2  5.802847 0.6493315 0.09816140 0.79910490 0.30320760 10.116929
3 15.045523 0.9683143 0.79207055 0.99040837 0.75895849 14.073861
4 20.458492 0.2172666 0.78221959 0.08559773 0.01160683 20.604627
5 11.436937 0.1583259 0.62951180 0.39425608 0.45021855 11.905628
6 15.195921 0.8157348 0.35058777 0.38213741 0.64069148 13.917235
```

The data contains the predictors `x0` to `x3` as well as the grouping factor `fac` indicating the nested structure of the data. We predict the response variable `y` using smooth terms of `x1` and `x2` and a varying intercept of `fac`. In addition, we assume the residual standard deviation `sigma` to vary by a smoothing term of `x0` and a varying intercept of `fac`.

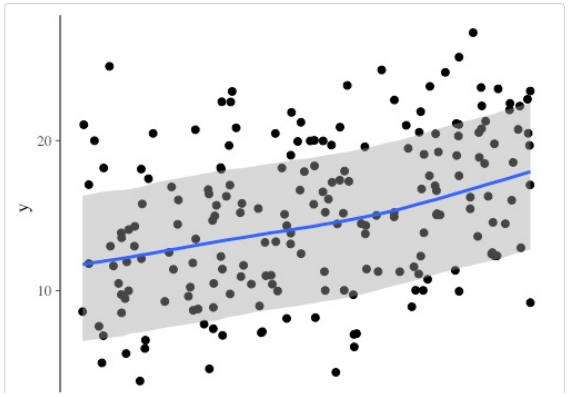
```
fit_smooth1 <- brm(
  bf(y ~ s(x1) + s(x2) + (1|fac), sigma ~ s(x0) + (1|fac)),
```

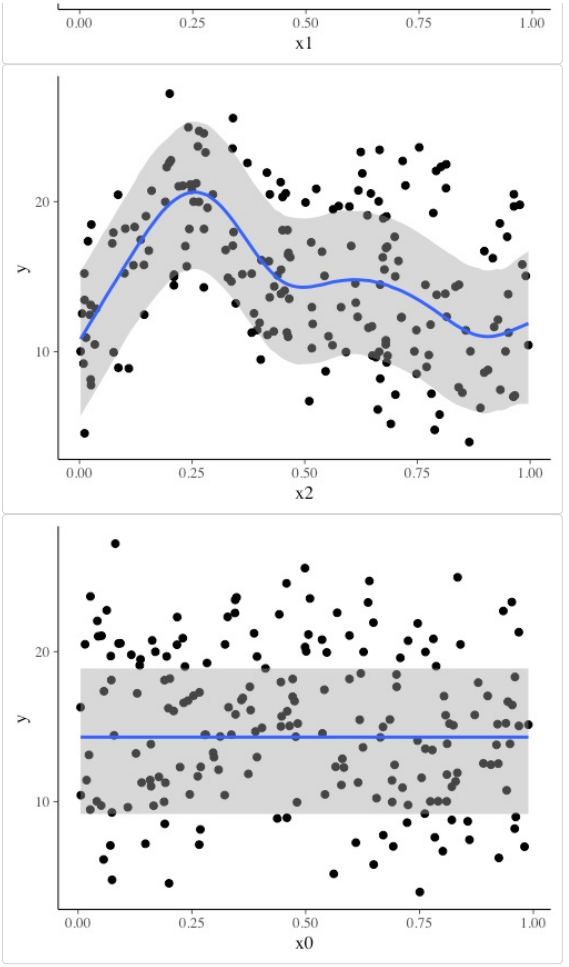
```
  data = dat_smooth, family = gaussian(),
  chains = 2, control = list(adapt_delta = 0.95)
)
```

```
summary(fit_smooth1)
```

```
 Family: gaussian
  Links: mu = identity; sigma = log
Formula: y ~ s(x1) + s(x2) + (1 | fac)
         sigma ~ s(x0) + (1 | fac)
   Data: dat_smooth (Number of observations: 200)
  Draws: 2 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 2000

Smooth Terms:
                Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sds(sx1_1)          1.90      1.47     0.13     5.63 1.00      957      951
sds(sx2_1)         19.60      5.49    11.91    32.89 1.00      741      931
sds(sigma_sx0_1)    1.66      1.29     0.11     4.85 1.00      614      955

Group-Level Effects:
~fac (Number of levels: 4)
                  Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)         5.06      2.36     2.30    11.19 1.00     1027     1417
sd(sigma_Intercept)   0.11      0.14     0.00     0.47 1.00      894     1059

Population-Level Effects:
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept        15.11      2.31    10.03    19.74 1.00     1104     1044
sigma_Intercept   0.67      0.09     0.50     0.87 1.00     1715     1194
sx1_1            11.29      3.94     3.92    20.33 1.00     1726     1263
sx2_1            49.36     12.88    24.30    74.77 1.00     1751     1160
sigma_sx0_1       1.17      3.28    -5.87     7.91 1.00     1555      988

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
plot(conditional_effects(fit_smooth1), points = TRUE, ask = FALSE)
```





This model is likely an overkill for the data at hand, but nicely demonstrates the ease with which one can specify complex models with **brms** and to fit them using **Stan** on the backend.