```
// generated with brms 2.20.1
functions {
}
data {
  int<lower=1> N;  // total number of observations
  vector[N] Y;  // response variable
  int<lower=-1,upper=2> cens[N];  // indicates censoring
  int<lower=1> K_alpha;  // number of population-level effects
  matrix[N, K_alpha] X_alpha;  // population-level design matrix
  int<lower=1> K_beta1;  // number of population-level effects
  matrix[N, K_beta1] X_beta1;  // population-level design matrix
  int<lower=1> K_beta2;  // number of population-level effects
  matrix[N, K_beta2] X_beta2;  // population-level design matrix
  int<lower=1> K_gamma;  // number of population-level effects
  matrix[N, K_gamma] X_gamma;  // population-level design matrix
  int<lower=1> K_kappa;  // number of population-level effects
  matrix[N, K_kappa] X_kappa;  // population-level design matrix
  // covariates for non-linear functions
  vector[N] C_1;
  // data for group-level effects of ID 1
  int<lower=1> N_1;  // number of grouping levels
  int<lower=1> M_1;  // number of coefficients per level
  int<lower=1> J_1[N];  // grouping indicator per observation
  // group-level predictor values
  vector[N] Z_1_alpha_1;
  // data for group-level effects of ID 2
  int<lower=1> N_2;  // number of grouping levels
  int<lower=1> M_2;  // number of coefficients per level
  int<lower=1> J_2[N];  // grouping indicator per observation
  // group-level predictor values
  vector[N] Z_2_alpha_1;
  // data for group-level effects of ID 3
  int<lower=1> N_3;  // number of grouping levels
  int<lower=1> M_3;  // number of coefficients per level
  int<lower=1> J_3[N];  // grouping indicator per observation
  // group-level predictor values
  vector[N] Z_3_beta1_1;
  // data for group-level effects of ID 4
  int<lower=1> N_4;  // number of grouping levels
  int<lower=1> M_4;  // number of coefficients per level
  int<lower=1> J_4[N];  // grouping indicator per observation
  // group-level predictor values
  vector[N] Z_4_beta1_1;
  // data for group-level effects of ID 5
  int<lower=1> N_5;  // number of grouping levels
  int<lower=1> M_5;  // number of coefficients per level
  int<lower=1> J_5[N];  // grouping indicator per observation
  // group-level predictor values
  vector[N] Z_5_beta2_1;
  // data for group-level effects of ID 6
  int<lower=1> N_6;  // number of grouping levels
  int<lower=1> M_6;  // number of coefficients per level
  int<lower=1> J_6[N];  // grouping indicator per observation
  // group-level predictor values
  vector[N] Z_6_beta2_1;
  // data for group-level effects of ID 7
  int<lower=1> N_7;  // number of grouping levels
  int<lower=1> M_7;  // number of coefficients per level
  int<lower=1> J_7[N];  // grouping indicator per observation
  // group-level predictor values
  vector[N] Z_7_gamma_1;
  // data for group-level effects of ID 8
  int<lower=1> N_8;  // number of grouping levels
  int<lower=1> M_8;  // number of coefficients per level
  int<lower=1> J_8[N];  // grouping indicator per observation
  // group-level predictor values
  vector[N] Z_8_kappa_1;
  int prior_only;  // should the likelihood be ignored?
}
transformed data {
}
parameters {
  vector[K_alpha] b_alpha;  // regression coefficients
  vector[K_beta1] b_beta1;  // regression coefficients
  vector[K_beta2] b_beta2;  // regression coefficients
  vector[K_gamma] b_gamma;  // regression coefficients
  vector[K_kappa] b_kappa;  // regression coefficients
  real<lower=0> sigma;  // dispersion parameter
  vector<lower=0>[M_1] sd_1;  // group-level standard deviations
  vector[N_1] z_1[M_1];  // standardized group-level effects
  vector<lower=0>[M_2] sd_2;  // group-level standard deviations
  vector[N_2] z_2[M_2];  // standardized group-level effects
  vector<lower=0>[M_3] sd_3;  // group-level standard deviations
  vector[N_3] z_3[M_3];  // standardized group-level effects
  vector<lower=0>[M_4] sd_4;  // group-level standard deviations
  vector[N_4] z_4[M_4];  // standardized group-level effects
  vector<lower=0>[M_5] sd_5;  // group-level standard deviations
  vector[N_5] z_5[M_5];  // standardized group-level effects
  vector<lower=0>[M_6] sd_6;  // group-level standard deviations
  vector[N_6] z_6[M_6];  // standardized group-level effects
  vector<lower=0>[M_7] sd_7;  // group-level standard deviations
  vector[N_7] z_7[M_7];  // standardized group-level effects
  vector<lower=0>[M_8] sd_8;  // group-level standard deviations
  vector[N_8] z_8[M_8];  // standardized group-level effects
}
transformed parameters {
  vector[N_1] r_1_alpha_1;  // actual group-level effects
  vector[N_2] r_2_alpha_1;  // actual group-level effects
  vector[N_3] r_3_beta1_1;  // actual group-level effects
  vector[N_4] r_4_beta1_1;  // actual group-level effects
  vector[N_5] r_5_beta2_1;  // actual group-level effects
  vector[N_6] r_6_beta2_1;  // actual group-level effects
  vector[N_7] r_7_gamma_1;  // actual group-level effects
  vector[N_8] r_8_kappa_1;  // actual group-level effects
  real lprior = 0;  // prior contributions to the log posterior
  r_1_alpha_1 = (sd_1[1] * (z_1[1]));
  r_2_alpha_1 = (sd_2[1] * (z_2[1]));
  r_3_beta1_1 = (sd_3[1] * (z_3[1]));
  r_4_beta1_1 = (sd_4[1] * (z_4[1]));
  r_5_beta2_1 = (sd_5[1] * (z_5[1]));
  r_6_beta2_1 = (sd_6[1] * (z_6[1]));
  r_7_gamma_1 = (sd_7[1] * (z_7[1]));
  r_8_kappa_1 = (sd_8[1] * (z_8[1]));
  lprior += student_t_lpdf(sigma | 3, 0, 2.5)
    - 1 * student_t_lccdf(0 | 3, 0, 2.5);
  lprior += student_t_lpdf(sd_1 | 3, 0, 2.5)
    - 1 * student_t_lccdf(0 | 3, 0, 2.5);
  lprior += student_t_lpdf(sd_2 | 3, 0, 2.5)
    - 1 * student_t_lccdf(0 | 3, 0, 2.5);
  lprior += student_t_lpdf(sd_3 | 3, 0, 2.5)
    - 1 * student_t_lccdf(0 | 3, 0, 2.5);
  lprior += student_t_lpdf(sd_4 | 3, 0, 2.5)
    - 1 * student_t_lccdf(0 | 3, 0, 2.5);
  lprior += student_t_lpdf(sd_5 | 3, 0, 2.5)
    - 1 * student_t_lccdf(0 | 3, 0, 2.5);
  lprior += student_t_lpdf(sd_6 | 3, 0, 2.5)
    - 1 * student_t_lccdf(0 | 3, 0, 2.5);
  lprior += student_t_lpdf(sd_7 | 3, 0, 2.5)
    - 1 * student_t_lccdf(0 | 3, 0, 2.5);
  lprior += student_t_lpdf(sd_8 | 3, 0, 2.5)
    - 1 * student_t_lccdf(0 | 3, 0, 2.5);
}
model {
  // likelihood including constants
  if (!prior_only) {
    // initialize linear predictor term
    vector[N] nlp_alpha = rep_vector(0.0, N);
    // initialize linear predictor term
    vector[N] nlp_beta1 = rep_vector(0.0, N);
    // initialize linear predictor term
    vector[N] nlp_beta2 = rep_vector(0.0, N);
    // initialize linear predictor term
    vector[N] nlp_gamma = rep_vector(0.0, N);
    // initialize linear predictor term
    vector[N] nlp_kappa = rep_vector(0.0, N);
    // initialize non-linear predictor term
    vector[N] mu;
    nlp_alpha += X_alpha * b_alpha;
    nlp_beta1 += X_beta1 * b_beta1;
    nlp_beta2 += X_beta2 * b_beta2;
    nlp_gamma += X_gamma * b_gamma;
    nlp_kappa += X_kappa * b_kappa;
    for (n in 1:N) {
      // add more terms to the linear predictor
      nlp_alpha[n] += r_1_alpha_1[J_1[n]] * Z_1_alpha_1[n] +
r_2_alpha_1[J_2[n]] * Z_2_alpha_1[n];
    }
    for (n in 1:N) {
      // add more terms to the linear predictor
      nlp_beta1[n] += r_3_beta1_1[J_3[n]] * Z_3_beta1_1[n] +
r_4_beta1_1[J_4[n]] * Z_4_beta1_1[n];
    }
    for (n in 1:N) {
      // add more terms to the linear predictor
      nlp_beta2[n] += r_5_beta2_1[J_5[n]] * Z_5_beta2_1[n] +
r_6_beta2_1[J_6[n]] * Z_6_beta2_1[n];
    }
    for (n in 1:N) {
      // add more terms to the linear predictor
      nlp_gamma[n] += r_7_gamma_1[J_7[n]] * Z_7_gamma_1[n];
    }
    for (n in 1:N) {
      // add more terms to the linear predictor
      nlp_kappa[n] += r_8_kappa_1[J_8[n]] * Z_8_kappa_1[n];
    }
    for (n in 1:N) {
      // compute non-linear predictor values
      mu[n] = (nlp_alpha[n] + nlp_beta1[n] * C_1[n] + nlp_beta2[n] *
nlp_gamma[n] * log((exp((C_1[n] - nlp_kappa[n]) / nlp_gamma[n]) +
exp( - (C_1[n] - nlp_kappa[n]) / nlp_gamma[n])) / (exp(nlp_kappa[n] /
nlp_gamma[n]) + exp( - nlp_kappa[n] / nlp_gamma[n]))));
    }
    for (n in 1:N) {
    // special treatment of censored data
      if (cens[n] == 0) {
        target += normal_lpdf(Y[n] | mu[n], sigma);
      } else if (cens[n] == 1) {
        target += normal_lccdf(Y[n] | mu[n], sigma);
      } else if (cens[n] == -1) {
        target += normal_lcdf(Y[n] | mu[n], sigma);
      }
    }
  }
  // priors including constants
  target += lprior;
  target += std_normal_lpdf(z_1[1]);
  target += std_normal_lpdf(z_2[1]);
  target += std_normal_lpdf(z_3[1]);
  target += std_normal_lpdf(z_4[1]);
  target += std_normal_lpdf(z_5[1]);
  target += std_normal_lpdf(z_6[1]);
  target += std_normal_lpdf(z_7[1]);
  target += std_normal_lpdf(z_8[1]);
}
generated quantities {
}
```