```
data {
      int<lower=0> N;
      // number of observations
      int<lower=0> Npat;
      // number of individuals
      real<lower=0> y[N];
      // outcome data
      real<lower=0> age[N];                                         //
explanatory variable (age) data
      int<lower=1,upper=Npat> id[N];                                //
patient id for each observation
      vector<lower=0,upper=0>[4] zeros4;                          // mean
vector for random effects distribution
      int<lower=0> betakp_lower;                                    //
lower bound for (prior on) mean knot point
      int<lower=0> betakp_upper;                                    //
upper bound for (prior on) mean knot point

      int<lower=0> Npred;
      // number of predicted observations
      int<lower=0> Npat_pred;
      // number of patients to predict observations for
      real<lower=0> age_pred[Npred];                                //
explanatory variable (age) for prediction
      int<lower=1,upper=Npat> id_pred[Npred];                     // patient
id for each predicted observation
}

parameters {
      vector[3] beta;
      // fixed effects, intercept and slopes
      real<lower=betakp_lower,upper=betakp_upper> betakp; // fixed
effects, knotpoint (bounding specified to help convergence)
      vector<lower=0>[4] u_sd;                                      //
level 2 error sd (sds of the random effects u[j])
      real<lower=0> y_sd;
      // level 1 error sd
      vector[4] u[Npat];
      // random effects (level 2) errors
      cholesky_factor_corr[4] L_u_Corr;                           //
cholesky factor for the random effects correlation matrix
}

transformed parameters {
      vector[4] alpha[Npat];                                        //
random effects
      real y_mu[N];
      // mean parameter based on regression equation

      //==========================
      // calculate random effects
      //==========================

      for (i in 1:Npat) {
            for (k in 1:3) alpha[i,k] <- beta[k] + u[i,k];
            alpha[i,4] <- betakp + u[i,4];
      }

      //=====================
      // regression equation
      //=====================

      for (j in 1:N) {
            if (age[j] < alpha[id[j],4])
                  y_mu[j] <- alpha[id[j],1] + alpha[id[j],2] * (age[j] -
alpha[id[j],4]);
            else
                  y_mu[j] <- alpha[id[j],1] + alpha[id[j],3] * (age[j] -
alpha[id[j],4]);
      }
}

model {

      //========
      // priors
      //========

      beta[1] ~ normal(20, 20);                                       //
prior: fixed effect, intercept
      beta[2] ~ normal(0, 4);
      // prior: fixed effect, slope before knot
      beta[3] ~ normal(0, 4);
      // prior: fixed effect, slope after knot
      betakp ~ uniform(betakp_lower,betakp_upper);       // prior: fixed
effect, knot point

      u_sd[1] ~ cauchy(0,5);                                          //
prior: random effect sd, intercept
      u_sd[2] ~ cauchy(0,5);                                          //
prior: random effect sd, slope before knot
      u_sd[3] ~ cauchy(0,5);                                          //
prior: random effect sd, slope after knot
      u_sd[4] ~ cauchy(0,5);                                          //
prior: random effect sd, knot point

      y_sd ~ cauchy(0,5);
      // prior: level 1 error sd

      L_u_Corr ~ lkj_corr_cholesky(1);                          // prior:
cholesky factor for random effects correlation matrix

            // NB. this prior is the "lkj correlation distribution" with
shape parameter 1

            // which is equivalent to a uniform distribution over the
possible correlation
```

```
            // matrices (where a shape parameter > 1 would have resulted
in an upside down

            // U-shaped distribution with the mode being located at the
identity matrix)

      //==============================
      // random effects distribution
      //==============================

      for (i in 1:Npat) u[i] ~ multi_normal_cholesky(zeros4,
diag_pre_multiply(u_sd, L_u_Corr));     // NB. the second parameter here
is the cholesky factor L

                                                                      //
(for the correlation matrix). It only uses the sd rather

                                                                      //
than the variances since Sigma = L*L'

      //==================
      // model likelihood
      //==================

      y ~ normal(y_mu, y_sd);
      // likelihood for the observed data

}

generated quantities {
      real y_pred[Npred];
      // predicted outcome
      real y_mu_pred[Npred];                                          //
predicted mean
      corr_matrix[4] u_Corr;                                          //
random effects correlation matrix
      matrix[4,4] u_Sigma;                                            //
random effects covariance matrix
      vector[4] alpha_tosave[Npat_pred];                        // monitor
random effects for a subset of patients only (for plotting predictions)
and do not monitor 'alpha' in the model above (since it consumes too much
memory!)

      //====================================================
      // predicted mean outcome using regression equation
      //====================================================

      for (i in 1:Npat_pred) {
            alpha_tosave[i] <- alpha[i];
      }

      for (j in 1:Npred) {
            if (age_pred[j] < alpha[id_pred[j],4])
```

```
                  y_mu_pred[j] <- alpha[id_pred[j],1] +
alpha[id_pred[j],2] * (age_pred[j] - alpha[id_pred[j],4]);
            else
                  y_mu_pred[j] <- alpha[id_pred[j],1] +
alpha[id_pred[j],3] * (age_pred[j] - alpha[id_pred[j],4]);


            y_pred[j] <- normal_rng(y_mu_pred[j], y_sd);
      }



      //=======================================================
      // recover the correlation and covariance matrices
      // using the cholesky factor of the correlation matrix
      //=======================================================

      u_Corr <- multiply_lower_tri_self_transpose(L_u_Corr);         //
correlation matrix: u_Corr = L_u_Corr * L_u_Corr'
      u_Sigma <- quad_form_diag(u_Corr, u_sd);
      // covariance matrix: u_Sigma = diag(u_sd) * u_Corr * diag(u_sd)

}
```