Stan demo with Spati data

In this notebook, we compare estimating linear mixed-effect models with NLME and Stan. The dataset comes from package "lmfor" and the example is the same that is used in mixed-effect model courses.

Preparing the data

```
library(lmfor)
data(spati)

# inlude only observations with measured growth
spati<-spati[spati$id2>0,]
spati$plot <- with(spati,factor(plot))
```

Estimation with NLME

The model estimates future tree growth based on its past growth. The tree measurements come from different plots that may have different growing conditions. The model assumes that the observations within one plot may be correlated with each other.

```
library(nlme)

spati_lme <- lme(id1~id2,random=~id2|plot,data=spati)
summary(spati_lme)
```

```
## Linear mixed-effects model fit by REML
##  Data: spati
##        AIC     BIC    logLik
##   26314.71 26353.5 -13151.36
##
## Random effects:
##  Formula: ~id2 | plot
##  Structure: General positive-definite, Log-Cholesky parametrization
##             StdDev   Corr
## (Intercept) 5.622427 (Intr)
## id2         0.173359 -0.578
## Residual    3.710059
##
## Fixed effects: id1 ~ id2
##                 Value Std.Error   DF   t-value p-value
## (Intercept) 1.8973172 0.7536626 4687  2.517462  0.0119
## id2         0.8059964 0.0256477 4687 31.425648  0.0000
##  Correlation:
##     (Intr)
## id2 -0.591
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -5.62386155 -0.46624008 -0.05002616  0.43340772 10.06902382
##
## Number of Observations: 4747
```

```
## Number of Groups: 59
```

BRMS package

Bayesian Regression Models with Stan (BRMS) uses Stan underneath, but offers lme4-like syntax for specifying the model

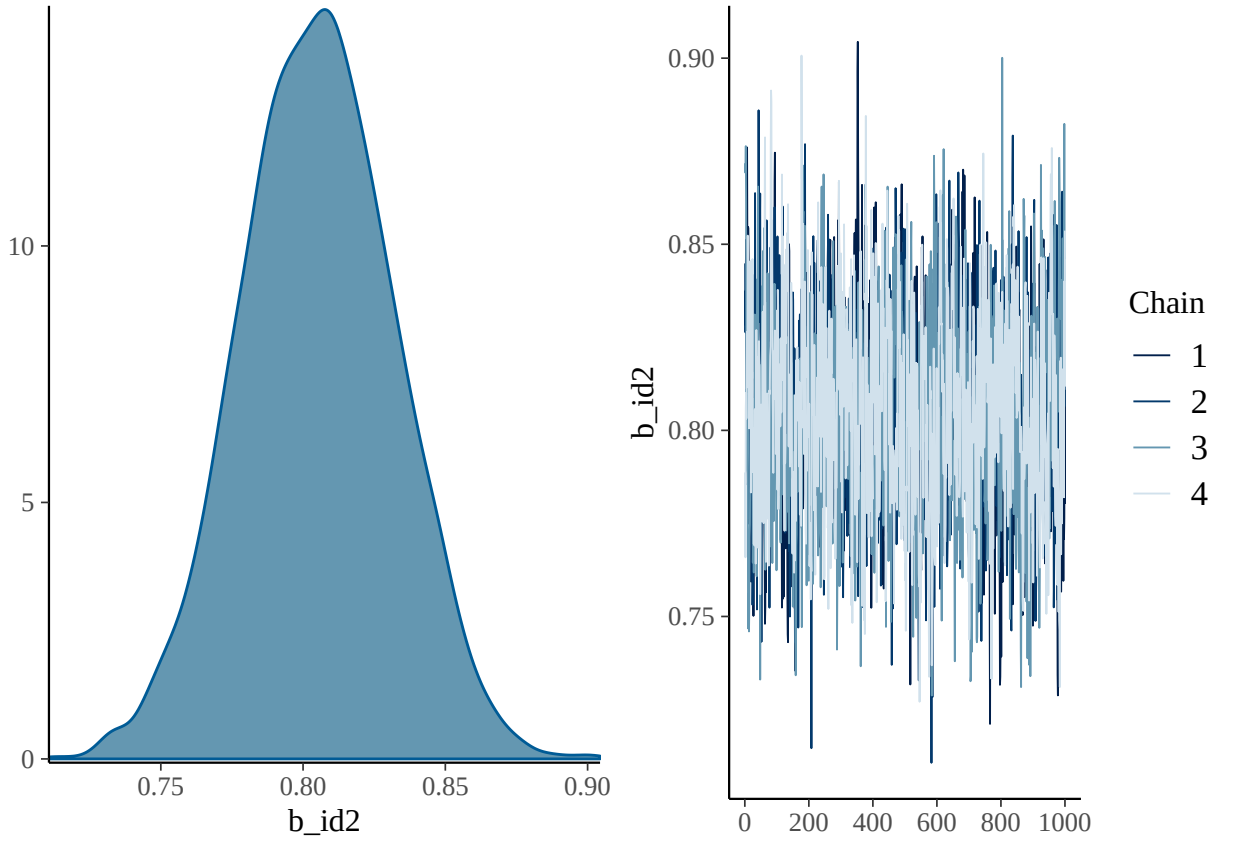
```
library(rstan)
library(brms)

# These options allow parallel execution of chains
rstan_options (auto_write=TRUE)
options (mc.cores=parallel::detectCores ())

# Estimation takes a while, so the estimated model object is cached to a file
modelfile <- "models/spati_brm.rds"
if (file.exists(modelfile))
{
  spati_brm <- readRDS(modelfile)
} else {
  spati_brm <- brm(formula = id1  ~ id2 + (1 + id2|plot),
                   data = spati, family = gaussian(),
                   warmup = 1000, iter = 2000, chains = 4,
                   control = list(adapt_delta = 0.95))

  saveRDS(spati_brm, file=modelfile)
}

spati_brm
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: id1 ~ id2 + (1 + id2 | plot)
##    Data: spati (Number of observations: 4747)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Group-Level Effects:
## ~plot (Number of levels: 59)
##                     Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sd(Intercept)           5.75      0.57     4.75     6.97       1343 1.00
## sd(id2)                 0.18      0.02     0.14     0.22       1502 1.00
## cor(Intercept,id2)     -0.56      0.09    -0.72    -0.36       1627 1.00
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept     1.89      0.77     0.36     3.38        427 1.00
## id2           0.81      0.03     0.75     0.86        905 1.00
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sigma     3.71      0.04     3.64     3.79       4000 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
```

```
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

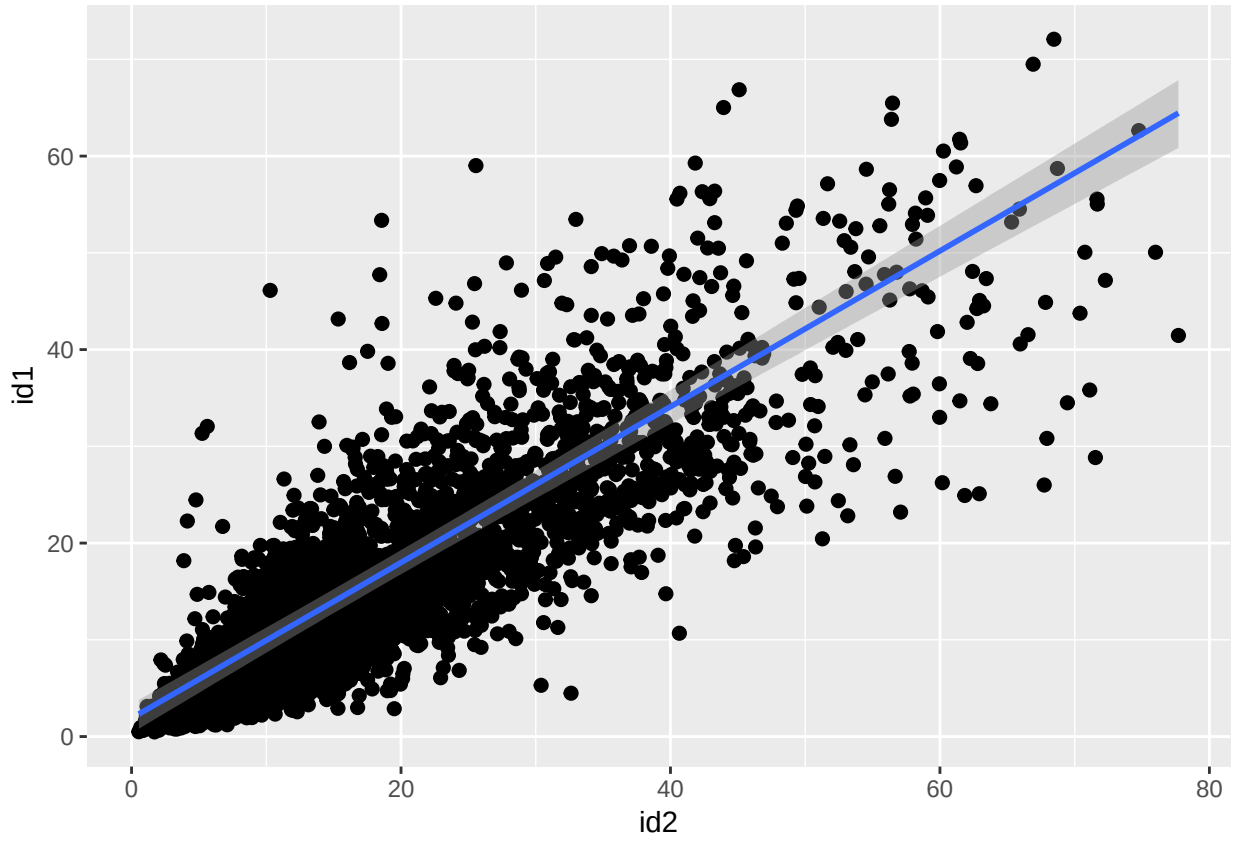
In comparison to the LME estimation, all the parameters have now posterior densities instead of point estimates. The BRM estimatation (0.81) of id2 coefficient matches the LME estimation, but now we can inspect its accuracy from the posterior.

```
plot(spati_brm, pars=c("b_id2"))
```



You can also inspect easily the marginal effect between past and future growth

```
plot(marginal_effects(spati_brm), points = TRUE)
```



BRMS generates Stan-code for the spesified regression model, and the generated code can be inspected with "stancode" method

```
#stancode(spati_brm)
```

RSTAN - Custom Stan-models

Next, we create the same linear mixed-effect model with hand crafted Stan-code where you can tweak all the parts of the model. The model itself is defined in a separate "spati.stan" files and here we only pass the data to it.

```
library(rstan)

# Parameters are feeded to Stan model as a list of values
params <- within(list(),
                 {
                   N <- nrow(spati)
                   Y <- as.vector(spati$id1)
                   X <- cbind(1,as.matrix(spati$id2))
                   Z <- X                                  # Z is indexed by 'group' so it's not block diago
                   p <- 2                                  # fixed-effects
                   k <- 2                                  # random-effects
                   J <- length(levels(spati$plot))
                   group <- as.integer(spati$plot)         # group index for data rows
                 })
```

```
modelfile <- "models/spati_stan.rds"
if (file.exists(modelfile))
{
  spati_stan <- readRDS(modelfile)
} else {

  rstan_options(auto_write=TRUE)
  options(mc.cores=parallel::detectCores())

  spati_stan <- stan(file="spati.stan", data=params, warmup=1000, iter=2000, chains=4, control = list(ad

  saveRDS(spati_stan, file=modelfile)
}
```

Once the model has been estimated we can do some inference on it with R. Let's make a sanity check that the Bayesian estimates are close to the LME (and BRM) estimates. Beta should be about 0.81.

```
print(spati_stan, pars = c("beta[1]", "beta_Intercept", "sigma_e", "sigma_b[1]", "sigma_b[2]"))
```

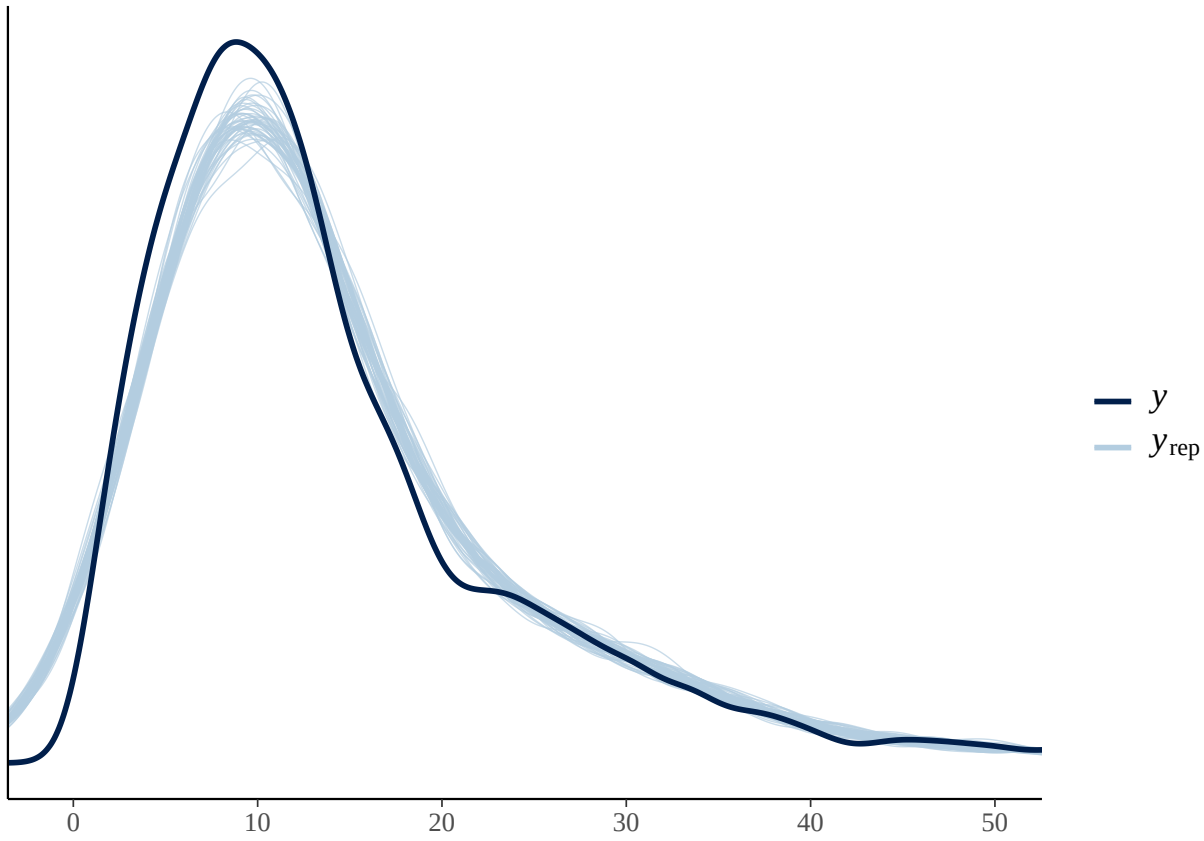
```
## Inference for Stan model: spati.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
## beta[1]        0.81    0.00 0.03 0.76 0.79 0.81 0.82  0.86  1347    1
## beta_Intercept 1.89    0.03 0.74 0.45 1.40 1.90 2.38  3.38   564    1
## sigma_e        3.71    0.00 0.04 3.64 3.69 3.71 3.74  3.78  4000    1
## sigma_b[1]     5.76    0.02 0.58 4.75 5.35 5.70 6.13  7.00  1472    1
## sigma_b[2]     0.18    0.00 0.02 0.14 0.16 0.18 0.19  0.22  1259    1
##
## Samples were drawn using NUTS(diag_e) at Tue Nov 20 08:40:40 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Posterior predictive check

Our custom Stan-code generates also a posterior predictive density where the in-sample values are feeded to the estimated model. If the model is accurate, it should generate a closely matching density with the original target values.

```
library(bayesplot)

posterior <- extract(spati_stan, pars = c("Y_rep"))
posterior_y_50 <- posterior$Y_rep[1:50,]

plot(ppc_dens_overlay(params$Y, posterior_y_50) +
       coord_cartesian(xlim = c(-1, 50)))
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```



Variational Bayes estimation

Instead of HMC sampling, Stan allows you to estimate the model also with variational bayes ADVI-algorithm that is significantly faster (but approximate)

```
library(rstan)

spati_model <- stan_model(file = "spati.stan")
spati_vb <- vb(spati_model, data=params, output_samples=2000, iter=3000, seed=678)
```

```
## ------------------------------------------------------------
## EXPERIMENTAL ALGORITHM:
##   This procedure has not been thoroughly tested and may be unstable
##   or buggy. The interface is subject to change.
## ------------------------------------------------------------
##
##
##
## Gradient evaluation took 0.001364 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 13.64 seconds.
## Adjust your expectations accordingly!
##
##
## Begin eta adaptation.
## Iteration:   1 / 250 [  0%]  (Adaptation)
## Iteration:  50 / 250 [ 20%]  (Adaptation)
```

```
## Iteration: 100 / 250 [ 40%]  (Adaptation)
## Iteration: 150 / 250 [ 60%]  (Adaptation)
## Iteration: 200 / 250 [ 80%]  (Adaptation)
## Success! Found best value [eta = 1] earlier than expected.
##
## Begin stochastic gradient ascent.
##   iter       ELBO   delta_ELBO_mean   delta_ELBO_med   notes
##    100     -3e+04             1.000            1.000
##    200     -2e+04             0.722            1.000
##    300     -1e+04             0.578            0.444
##    400     -1e+04             0.247            0.290
##    500     -1e+04             0.100            0.008   MEDIAN ELBO CONVERGED
##
## Drawing a sample of size 2000 from the approximate posterior...
## COMPLETED.
```

```
print(spati_vb, pars = c("beta[1]", "beta_Intercept", "sigma_e", "sigma_b[1]", "sigma_b[2]"))
```

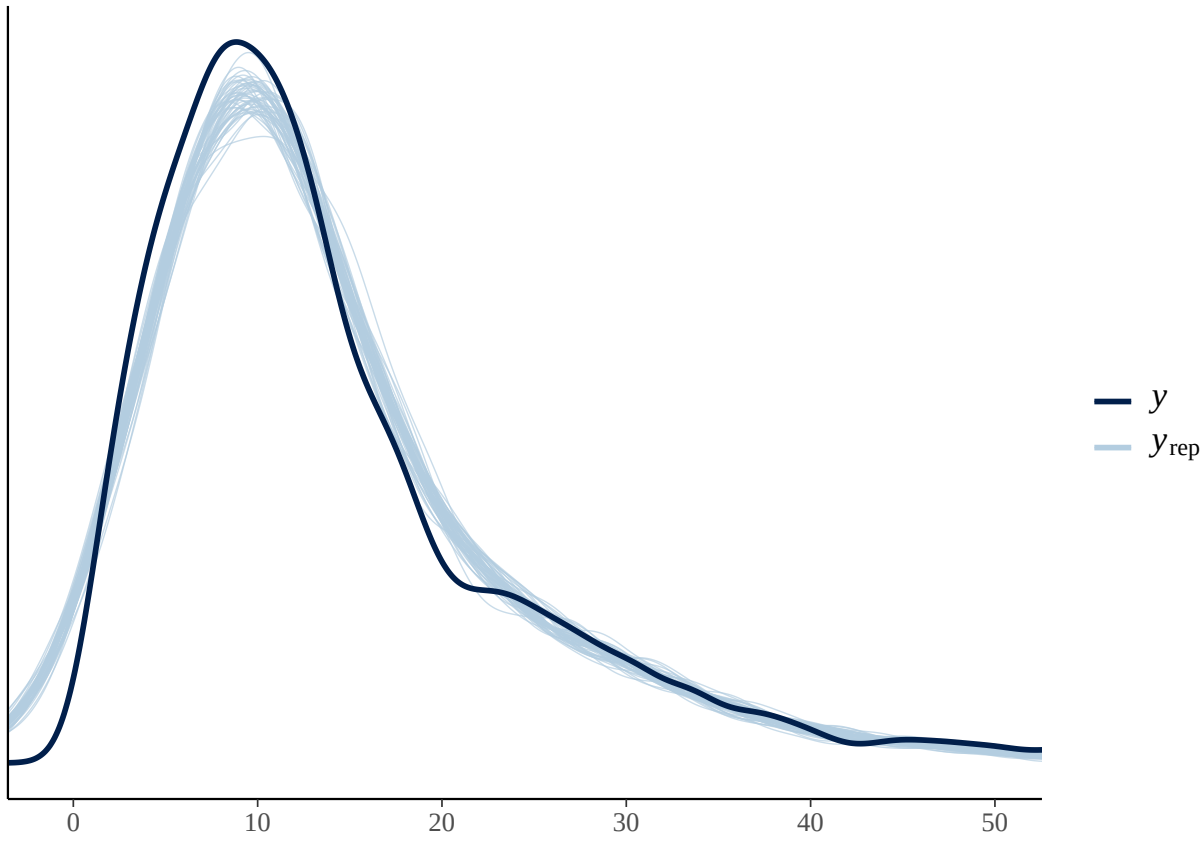
```
## Inference for Stan model: spati.
## 1 chains, each with iter=2000; warmup=0; thin=1;
## post-warmup draws per chain=2000, total post-warmup draws=2000.
##
##                 mean   sd  2.5%   25%   50%   75% 97.5%
## beta[1]         0.82 0.00  0.81  0.82  0.82  0.82  0.82
## beta_Intercept -4.15 0.08 -4.31 -4.21 -4.15 -4.10 -3.99
## sigma_e         3.77 0.05  3.68  3.74  3.77  3.81  3.87
## sigma_b[1]      7.61 0.09  7.44  7.55  7.61  7.66  7.77
## sigma_b[2]      0.13 0.00  0.13  0.13  0.13  0.13  0.14
##
## Approximate samples were drawn using VB(meanfield) at Tue Nov 20 12:48:31 2018.
```

```
## We recommend genuine 'sampling' from the posterior distribution for final inferences!
```

The estimate of beta (0.82) is quite close tio HMC estimate (0.81). Let's finally check the predicted density of VB model..

```
library(bayesplot)

posterior <- extract(spati_vb, pars = c("Y_rep"))
posterior_y_50 <- posterior$Y_rep[1:50,]

plot(ppc_dens_overlay(params$Y, posterior_y_50) +
       coord_cartesian(xlim = c(-1, 50)))
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```


0
10
20
30
40
50
y
y rep