Bayesian Correlated Individual Hetereogeneity Modelling

July 2024

Importing the data

We begin by loading the required packages:

```
library(tidyverse)
library(rstan)
library(HDInterval)
library(bayesplot)
library(ggplot2)
library(truncnorm)
```

We now import the data files: after storing the path to the working directory (WD) in the `work_dir` object, we can set the WD and load the auxiliary functions:

```
setwd(work_dir)
```

```
dat <- read.csv("panel_fadn.csv")
source("aux_functions.R")
```

This dataset contains a balanced panel of Italian farms extracted from the 2015-2020 Italian Farm Agricultural Data Network (FADN). It consists of five farm-level covariates:

- Altitude: `code_alt`
- Arable land: `cult_land`
- Forest Land: `forest_land`
- Machinery total power: `kw_machinery`
- Number of hired workers: `workers`

two year/ID indicators:

- Farm ID: `id`
- Year: `year`

and the target outcome, Net Farm Income (NFI): `earn`.

```
summary(dat)
```

```
##        id              year         code_alt       cult_land
##  Min.   :  1.0   Min.   :2015   Min.   :1.000   Min.   :  0.100
##  1st Qu.:105.8   1st Qu.:2016   1st Qu.:2.000   1st Qu.:  6.107
```

```
##  Median :210.5   Median :2018   Median :2.000   Median : 13.000
##  Mean   :210.5   Mean   :2018   Mean   :2.092   Mean   : 30.271
##  3rd Qu.:315.2   3rd Qu.:2019   3rd Qu.:3.000   3rd Qu.: 33.320
##  Max.   :420.0   Max.   :2020   Max.   :3.000   Max.   :653.800
##   forest_land       kw_machinery       workers           earn
##  Min.   :  0.000   Min.   :   9.0   Min.   :0.250   Min.   :   0.071
##  1st Qu.:  0.000   1st Qu.:  66.0   1st Qu.:0.910   1st Qu.:   9.953
##  Median :  0.000   Median : 117.0   Median :1.000   Median :  23.644
##  Mean   :  3.418   Mean   : 184.9   Mean   :1.338   Mean   :  53.143
##  3rd Qu.:  0.200   3rd Qu.: 213.0   3rd Qu.:1.710   3rd Qu.:  57.391
##  Max.   :460.580   Max.   :1961.0   Max.   :5.360   Max.   :1394.532
```

Modelling Strategy

Given the structure of the data, one sensible strategy to model the association between NFI and the four independent variables is to devise a strategy to control for the potential association between individual additive heterogeneity α_i and the covariates $\mathbf{x}_{it}$. The classical econometric tool that allows to address this kind of identification problem is the *within* estimator, also known as *fixed effect estimator*. This is typically defined as follows: consider the model

$$y_{it} = \alpha_0 + \alpha_i + \sum_{p=1}^{P} \beta_p x_{it,p} + \varepsilon_{it} \tag{1}$$

were $\mathbb{E}[\alpha_i x_{it,p}] \neq 0$ for any p. Then a constent estimator for β_p is given be the regressing y_{it} on $\dot{x}_{it,p}$, where $\dot{x}_{it,p} = x_{it,p} - \bar{x}_{i,p}$ and $\bar{x}_{i,p} = T_i^{-1} \sum_{t=1}^{T} x_{it,p}$. Although this is an easy and popular way to get rid of correlated heterogeneity, most of the statistical literature now agrees that the price to pay in terms of information loss is sometimes unjustified, particularly in the face of concurrent methods that allow achieving the same result at a much lower price. For example, Chamberlain (1982) showed that an analogous estimator can be obtained by augmenting equation (1) with a set of secondary regressors:

$$y_{it} = \alpha_0 + \alpha_i + \sum_{p=1}^{P} \beta_p x_{it,p} + \sum_{p=1}^{P} \sum_{t=1}^{T} \gamma_{tp} x_{i,tp} + \epsilon_{it} \tag{2}$$

A similar strategy was also discussed by Mundlak (1978), who proposed substituting $\sum_{p=1}^{P} \sum_{t=1}^{T} \gamma_p x_{it,p}$ with $\sum_{p=1}^{P} \gamma_p \tilde{x}_{i,p}$, where $\tilde{x}_{i,p} = T^{-1} \sum_{t=1}^{T} x_{it,p}$. From a standard econometric perspective, both model (2) and the Mundlak equivalent can be estimated more efficiently than with a standard within estimator by adopting a RE specification. In other words, by assuming

$$\alpha_i \sim \mathrm{N}(0, \sigma_\alpha)$$

and estimating β_p via Generalized Least Squares (GLS), the resulting estimator β_p^{RE} will be more efficient than β_p^{within}.

From a Bayesian perspective, the Chamberlain approach to correlated individual heterogeneity is particularly appealing as it naturally extends the representation of the potential data generating process, rather than relying on data transformation. Indeed, one can think model (2) as simply giving α_i a *multilevel* (or *hierarchical*) prior:

$$
\begin{aligned}
y_{it} &\sim \mathrm{N}(\mu_y, \sigma_y) \\
\sigma_y &\sim \mathrm{N}_+(0, \rho_y) \\
\mu_y &= \alpha_0 + \alpha_i + \sum_{p=1}^{P} \beta_p x_{it,p} \\
\alpha_0 &\sim \mathrm{N}(0, \sigma_{\alpha_0}) \\
\beta_p &\sim \mathrm{N}(0, \sigma_\beta) \\
\alpha_i &\sim \mathrm{N}(\mu_i, \sigma_\alpha) \\
\mu_i &= \sum_{p=1}^{P} \sum_{t=1}^{T} \gamma_{tp} x_{i,tp} \\
\gamma_{tp} &\sim \mathrm{N}(0, \sigma_\gamma) \\
\sigma_\alpha &\sim \mathrm{N}_+(0, \rho_\alpha)
\end{aligned}
\tag{3}
$$

This structure extends the Normal linear regression model by adding α_i to the conditional mean function. Then, the distributional assumptions for this additive term (i.e., its mean and variance) regulate its dependence with $\mathbf{x}_{it}$, thereby solving the correlation problem. The flip side of having such a complex setup is choosing sensible hyperparameters for the priors in equation (3). Whereas choosing which distribution to give α_0 and β_p follows the same logic discussed in the case of standard regression, things get more complicated for σ_α and γ_{tp} as their impact on $y_{i,t}$ is less straightforward.

Data Preparation and Prior calibration

Before exploring prior calibration in case of panel data models, we make some adjustment to our initial dataset:

```
dat_tr <- dat %>%
  mutate(forest_land_sh = forest_land / cult_land,
         earn_ha = earn / cult_land,
         kw_machinery_ha = kw_machinery / cult_land,
         workers_ha = workers / cult_land)
```

It should be clear by now that the best way to approach priors in Bayesian modelling is to begin by standardizing either the outcome, the regressors or both. Althought the consequences of such operation are obvious in the case of standard linear regression, things can get more complicated in multilevel models such as the Chamberlain device. For example, in a setting where each variable incorporates both individual and time variation, standardization can be performed with respect to the *grand mean/sd*, the *individual mean/sd* or the *time mean/sd*. These means can be defines as:

$$
\begin{aligned}
\bar{z} &= NT^{-1} \sum_{t=1}^{T} \sum_{i=1}^{N} z_{it} \\
\bar{z}_i &= T^{-1} \sum_{t=1}^{T} z_{it} \\
\bar{z}_t &= N^{-1} \sum_{i=1}^{N} z_{it}
\end{aligned}
$$

while the standard deviations are computed through the differences between z_{it} the corresponding mean. Clearly, each choice leads to a different estimate of β_p which requires transforming the coefficient back to

data scale. For simplicity and consistency with the previous applications, we will standardize our data using the grand mean and grand standard deviation.

```
X <- dat_tr %>%
  select(cult_land, forest_land_sh, kw_machinery_ha, workers_ha) %>%
  mutate_all(scale) %>%
  as.matrix()

U_chmb <- dat_tr %>%
  select(year, cult_land, forest_land_sh, kw_machinery_ha, workers_ha) %>%
  group_by(year) %>%
  pivot_wider(names_from = year,
              values_from = cult_land:workers_ha) %>%
  unnest(everything()) %>%
  mutate_all(scale) %>%
  as.matrix()

std_earn <- as.vector(scale(dat$earn))
```

As we have seen in the standard linear regression setup, this operation forces $\alpha_0 = 0$. Therefore, we once more begin by studying a reasonable prior for σ_y. Thanks to standardization, we can easily start at $\sigma_y \sim \mathrm{N}_+(0, 1)$. First, load all the necessary `Stan` files:

```
rstan_options(auto_write = TRUE)
chmb_prior <- stan_model("chamberlain_regression_prior_1.stan")
chmb_prior_re <- stan_model("chamberlain_regression_prior_2.stan")
chmb_prior_re_sl <- stan_model("chamberlain_regression_prior_3.stan")
chmb_prior_re_sl_b <- stan_model("chamberlain_regression_prior_4.stan")
```

Then, simulate from $\sigma_y \sim \mathrm{N}_+(0, 1)$ and the PrPD of y_{it}:

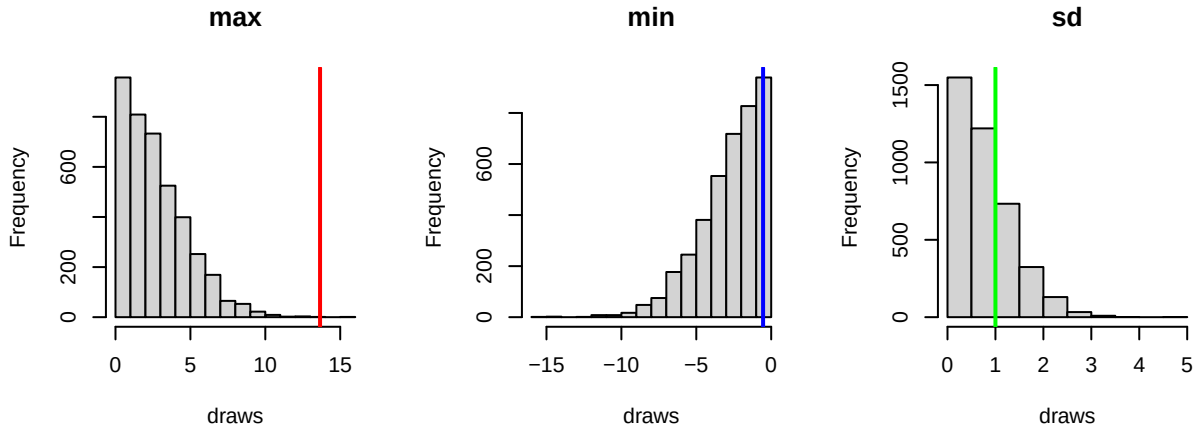
```
dat_list <- list(
  y = std_earn,
  X = X,
  U = U_chmb,
  J = length(unique(dat$id)),
  N = nrow(dat),
  K = ncol(X),
  L = ncol(U_chmb),
  jj = dat$id,
  sd_sigma_y = 1,
  sd_inter = 0,
  mu_inter = 0
)

fit_prior <- sampling(chmb_prior,
                      data = dat_list,
                      algorithm = "Fixed_param")
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Sampling)
```

```
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.308 seconds (Sampling)
## Chain 1:                0.308 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0 seconds (Warm-up)
## Chain 2:                0.31 seconds (Sampling)
## Chain 2:                0.31 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0 seconds (Warm-up)
## Chain 3:                0.311 seconds (Sampling)
## Chain 3:                0.311 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Sampling)
```

```
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0 seconds (Warm-up)
## Chain 4:                0.312 seconds (Sampling)
## Chain 4:                0.312 seconds (Total)
## Chain 4:
```

```r
fit_prior_smpl <- extract(fit_prior)
y_sim <- fit_prior_smpl$y

par(mfrow=c(1,3), pty="s")
hist(apply(y_sim, 1, max), main = "max", xlab = "draws")
abline(v=max(std_earn),col="red", lwd=2)
hist(apply(y_sim, 1, min), main = "min", xlab = "draws")
abline(v=min(std_earn),col="blue", lwd=2)
hist(apply(y_sim, 1, sd), main = "sd", xlab = "draws")
abline(v=sd(std_earn),col="green", lwd=2)
```




```
par(mfrow=c(1,1), pty="m")
```

This prior looks fine overall, although the generated maximum values are well below the observed one.

Prior on β_p

As already mentioned, given the way we standardized our outcome and covariates, it also make sense to set $\beta_p \sim \mathrm{N}(0, 1)$ as we did in the linear regression model case study:

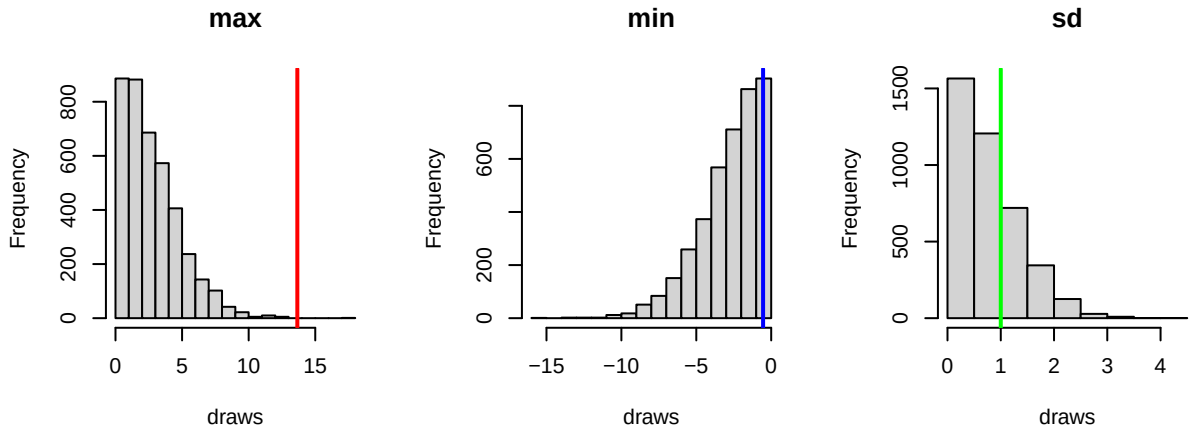
```
dat_list$sd_beta <- 1

fit_prior <- sampling(chmb_prior,
                      data = dat_list,
                      algorithm = "Fixed_param")
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Sampling)
```

```
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.309 seconds (Sampling)
## Chain 1:                0.309 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0 seconds (Warm-up)
## Chain 2:                0.31 seconds (Sampling)
## Chain 2:                0.31 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0 seconds (Warm-up)
## Chain 3:                0.31 seconds (Sampling)
## Chain 3:                0.31 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Sampling)
```

```
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0 seconds (Warm-up)
## Chain 4:                0.318 seconds (Sampling)
## Chain 4:                0.318 seconds (Total)
## Chain 4:
```

```r
fit_prior_smpl <- extract(fit_prior)
y_sim <- fit_prior_smpl$y

par(mfrow=c(1,3), pty="s")
hist(apply(y_sim, 1, max), main = "max", xlab = "draws")
abline(v=max(std_earn),col="red", lwd=2)
hist(apply(y_sim, 1, min), main = "min", xlab = "draws")
abline(v=min(std_earn),col="blue", lwd=2)
hist(apply(y_sim, 1, sd), main = "sd", xlab = "draws")
abline(v=sd(std_earn),col="green", lwd=2)
```



```
par(mfrow=c(1,1), pty="m")
```

Again, this configuration seems reasonable, although the observed maximum values still fall in the right tail of the PrPD. However, this may be hard to directly address while keeping the prior minimum and the prior standard deviation coherent with the observed ones. Indeed, the distribution of NFI is right-skewed and a Normal model for y_{it} might not be the best choice. As we will see later on, however, we can attempt to circumvent this problem through the individual heterogeneity components.

Prior on individual heterogeneity

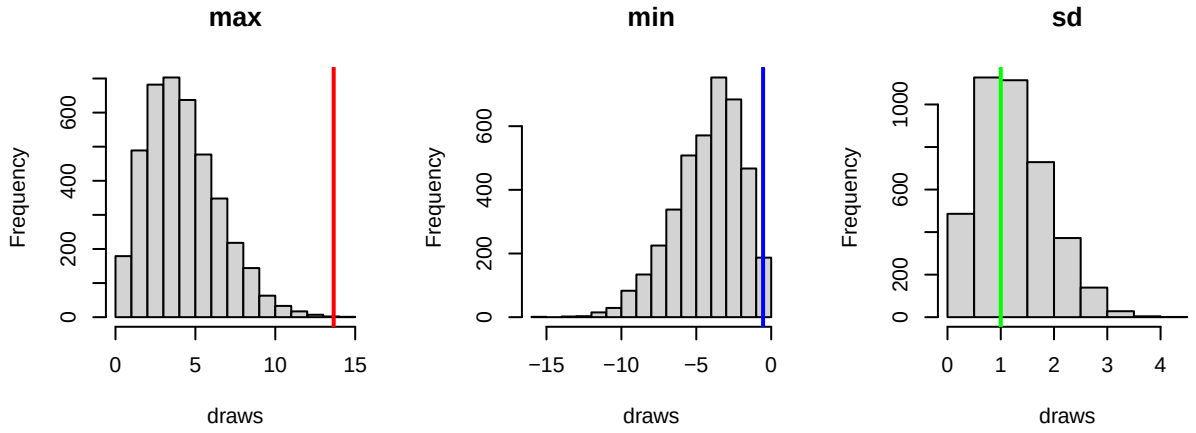
For the time being, let us assume that $\gamma_{tp} = 0$ for all p and t such that $\alpha_i \sim \mathrm{N}(0, \sigma_\alpha)$. Since $\mathrm{sd}(\mathbf{y}) = 1$ it make sense to set $\sigma_\alpha \sim \mathrm{N}_+(0, 1)$:

```
dat_list$sd_sigma_a <- 1

fit_prior_re <- sampling(chmb_prior_re,
                         data = dat_list,
                         algorithm = "Fixed_param")
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.442 seconds (Sampling)
## Chain 1:                0.442 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0 seconds (Warm-up)
```

```
## Chain 2:                0.441 seconds (Sampling)
## Chain 2:                0.441 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0 seconds (Warm-up)
## Chain 3:                0.444 seconds (Sampling)
## Chain 3:                0.444 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0 seconds (Warm-up)
## Chain 4:                0.445 seconds (Sampling)
## Chain 4:                0.445 seconds (Total)
## Chain 4:
```

```
fit_prior_smpl <- extract(fit_prior_re)
y_sim <- fit_prior_smpl$y

par(mfrow=c(1,3), pty="s")
hist(apply(y_sim, 1, max), main = "max", xlab = "draws")
abline(v=max(std_earn),col="red", lwd=2)
hist(apply(y_sim, 1, min), main = "min", xlab = "draws")
abline(v=min(std_earn),col="blue", lwd=2)
hist(apply(y_sim, 1, sd), main = "sd", xlab = "draws")
abline(v=sd(std_earn),col="green", lwd=2)
```




```
par(mfrow=c(1,1), pty="m")
```

This setting seems reasonable as well, although maximum values are not quite there yet. Clearly, being NFI asymmetric, tweaking the additive **zero-mean** individual heterogeneity and σ_y can only help so much. What happens when we consider the individual covariates in μ_i?

Prior on γ_{tp}

We now introduce a model for the conditional mean of α_i, μ_i. Using the Chamberlain approach, for $t_0 = 2015$ and $T = 2020$, we need to include $P* 6$ covariates and that many coefficients. When either P or T (or both) are large, this modelling approach can lead to overfit and, as a consequence, (very) high variance. One way to mitigate this issue is called *regularization*. Regularized regression is a generic term that refers to a wide set of modelling techniques aimed at tackling overfit by forcing some of the coefficient to either zero or small values. Although regularized estimates of γ_{tp} will be biased by construction, if these are not of direct interest to the researcher, one can use such techniques to reduce the complexity of μ_i. The so-called *soft-regularization* (i.e., penalties that do not force coefficients to exactly zero) can be easily implemented in any Bayesian regression model by simply imposing a prior on the variance of the slope terms:

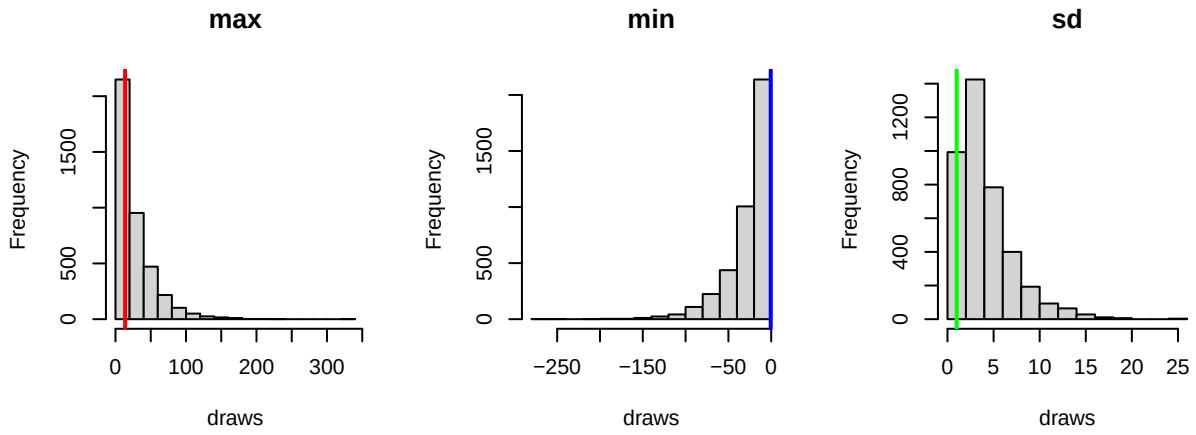
$$\begin{aligned} \gamma_{tp} &\sim \mathrm{N}(0, \sigma_\gamma) \\ \sigma_\gamma &\sim \mathrm{N}_+(0, \rho_\gamma) \\ \rho_\gamma &= 1 \end{aligned} \tag{1}$$

```r
dat_list$sd_sigma_g <- 1

fit_prior_re_sl <- sampling(chmb_prior_re_sl,
                            data = dat_list,
                            algorithm = "Fixed_param")
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.457 seconds (Sampling)
## Chain 1:                0.457 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0 seconds (Warm-up)
## Chain 2:                0.46 seconds (Sampling)
## Chain 2:                0.46 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
```

```
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0 seconds (Warm-up)
## Chain 3:                0.46 seconds (Sampling)
## Chain 3:                0.46 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0 seconds (Warm-up)
## Chain 4:                0.459 seconds (Sampling)
## Chain 4:                0.459 seconds (Total)
## Chain 4:
```

```r
fit_prior_smpl <- extract(fit_prior_re_sl)
y_sim <- fit_prior_smpl$y

par(mfrow=c(1,3), pty="s")
hist(apply(y_sim, 1, max), main = "max", xlab = "draws")
abline(v=max(std_earn),col="red", lwd=2)
hist(apply(y_sim, 1, min), main = "min", xlab = "draws")
abline(v=min(std_earn),col="blue", lwd=2)
hist(apply(y_sim, 1, sd), main = "sd", xlab = "draws")
abline(v=sd(std_earn),col="green", lwd=2)
```




```
par(mfrow=c(1,1), pty="m")
```

Introducing the term $\sum_{p=1}^{P}\sum_{t=1}^{T}\gamma_{tp}x_{i,tp}$ make the PrPD leptokurtic (i.e., it thickens the tail of the starting normal distribution model). While this is a desirable outcome when it comes to extend its potential maximum values, minimums and SDs are also affected. Indeed, the range of plausible minimum NFI values under the current prior configuration plunges to roughly -250, which is clearly unreasonably small. Because of symmetry, maximum values now extend to up to more then 200 as well. We can try to reduce the range of the PrPD by setting a lower value of ρ_γ:

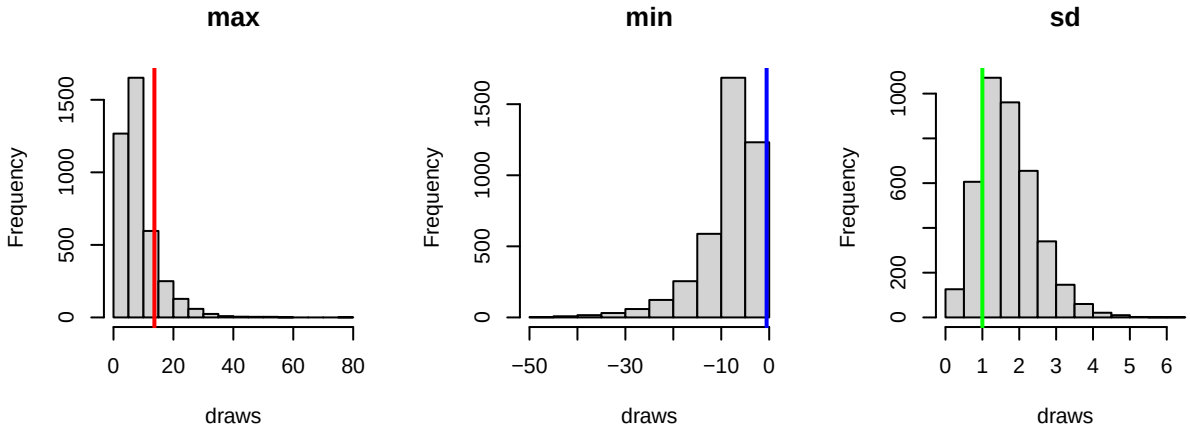
```
dat_list$sd_sigma_g <- .25

fit_prior_re_sl <- sampling(chmb_prior_re_sl,
                            data = dat_list,
                            algorithm = "Fixed_param")
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Sampling)
```

```
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.455 seconds (Sampling)
## Chain 1:                0.455 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0 seconds (Warm-up)
## Chain 2:                0.456 seconds (Sampling)
## Chain 2:                0.456 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Sampling)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0 seconds (Warm-up)
## Chain 3:                0.461 seconds (Sampling)
## Chain 3:                0.461 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4: Iteration:    1 / 2000 [  0%]  (Sampling)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Sampling)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Sampling)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Sampling)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Sampling)
```

```
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0 seconds (Warm-up)
## Chain 4:                0.456 seconds (Sampling)
## Chain 4:                0.456 seconds (Total)
## Chain 4:
```

```r
fit_prior_smpl <- extract(fit_prior_re_sl)
y_sim <- fit_prior_smpl$y

par(mfrow=c(1,3), pty="s")
hist(apply(y_sim, 1, max), main = "max", xlab = "draws")
abline(v=max(std_earn),col="red", lwd=2)
hist(apply(y_sim, 1, min), main = "min", xlab = "draws")
abline(v=min(std_earn),col="blue", lwd=2)
hist(apply(y_sim, 1, sd), main = "sd", xlab = "draws")
abline(v=sd(std_earn),col="green", lwd=2)
```



```
par(mfrow=c(1,1), pty="m")
```

Limitations of PPC in multilevel models

Clearly, our calibration exercise has several limitations. On the one hand, choosing $y_{i,t} \sim \mathbf{N}(\cdot, \cdot)$ might be not a good idea as the unstandardized NFI is clearly restricted to positive values with a markedly pronounced right tail. In such situations, there are other classes of distribution that can easily replace the normal model: these include the **Gamma** distribution and the **log-Normal** distribution. Secondly, we approached prior calibration by pooling all individuals and years and tackling the resulting PrPD without looking into specific individuals/years with potentially deviating behavior. Although the latter is a much better and more precise way to determine the shape and scale of the prior ensemble, it both is more difficult to implement and time consuming (in terms of coding and number of tuning parameters). Therefore, the next section discusses estimation conditional on the calibration exercise conducted so far. We leave the two previously mentioned extensions for future discussions.

Estimation

Estimating the Chamberlain model follows the same procedure discussed for both Normal and Logistic regression models:

```
dat_list <- list(
  y = std_earn,
  X = X,
  U = U_chmb,
  J = length(unique(dat$id)),
  N = nrow(dat),
  K = ncol(X),
  L = ncol(U_chmb),
  jj = dat$id,
  sd_inter = 0,
  mu_inter = 0,
  sd_sigma_a = 1,
  sd_sigma_y = 1,
  sd_sigma_g = .25,
  sd_beta = 1
)

chmb <- stan_model("chamberlain_regression.stan")

fit <- sampling(chmb,
                data = dat_list,
                chains = 4,
                cores = 4)
```

Compare the estimated coefficients with those obtain using a fixed effect model:

```
sample_fit <- extract(fit)
beta <- sample_fit$beta

fe_reg <- plm::plm(earn ~ cult_land + forest_land_sh + kw_machinery_ha + workers_ha,
```

```
                  data = dat_tr %>%
                    mutate_at(vars(earn, cult_land, forest_land_sh,
                                   kw_machinery_ha, workers_ha), scale))

t(round(apply(beta, 2, quantile, c(.025, .975)), 3))
```

```
##
##          2.5% 97.5%
##   [1,]  0.295 0.541
##   [2,]  0.059 0.387
##   [3,] -0.135 0.049
##   [4,] -0.075 0.153
```

```
round(confint(fe_reg), 3)
```

```
##                   2.5 % 97.5 %
## cult_land         0.291  0.552
## forest_land_sh    0.060  0.401
## kw_machinery_ha -0.138  0.047
## workers_ha      -0.074  0.159
```

Since $\mathbf{x}_{it}$ and y_{it} have been standardized, we need to map the estimated coefficients back to the original scale of the data:

```
X_unsc <- dat_tr %>%
  select(cult_land, forest_land_sh, kw_machinery_ha, workers_ha) %>%
  as.matrix()

sc_factor <- sd(dat$earn)/apply(X_unsc, 2, sd)

beta_backsc <- sweep(beta, 2, sc_factor, "*")

fe_reg_unsc <- plm::plm(earn ~ cult_land + forest_land_sh + kw_machinery_ha +
                          workers_ha, data = dat_tr)

t(round(apply(beta_backsc, 2, quantile, c(.025, .975)), 3))
```

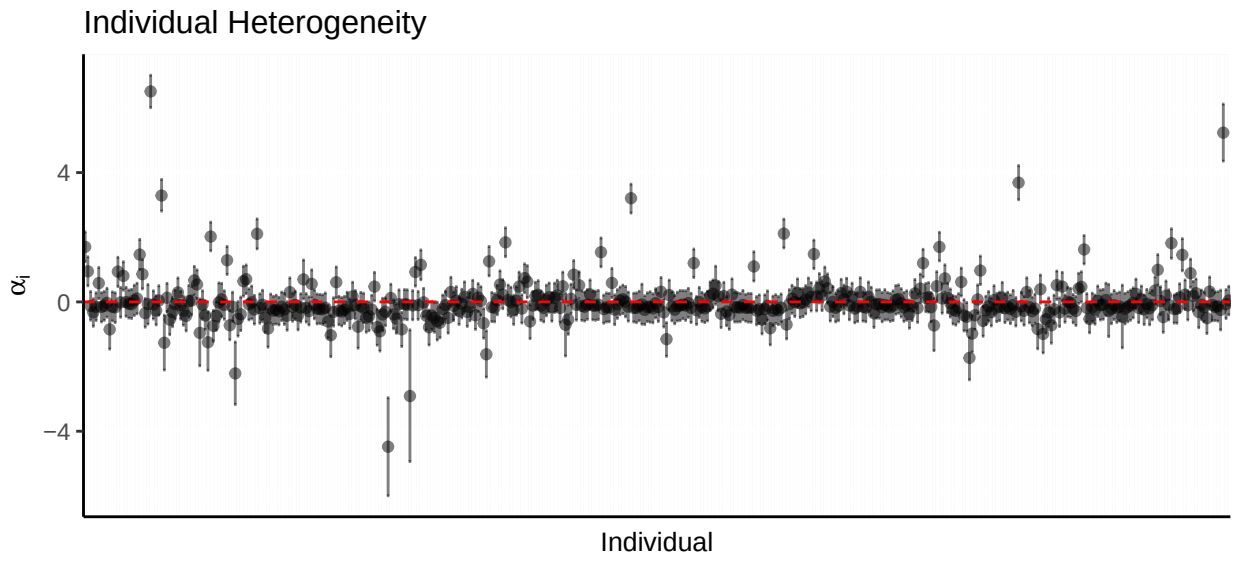
```
##
##           2.5%   97.5%
##   [1,]   0.590   1.081
##   [2,]  16.641 109.889
##   [3,]  -0.500   0.182
##   [4,] -11.555  23.600
```

```
round(confint(fe_reg_unsc), 3)
```

```
##                    2.5 %  97.5 %
## cult_land          0.582   1.103
## forest_land_sh    16.998 113.985
## kw_machinery_ha  -0.510   0.174
## workers_ha      -11.449  24.576
```

We can also visualize the estimated additive heterogeneity, i.e., the individual components α_i in model (3):

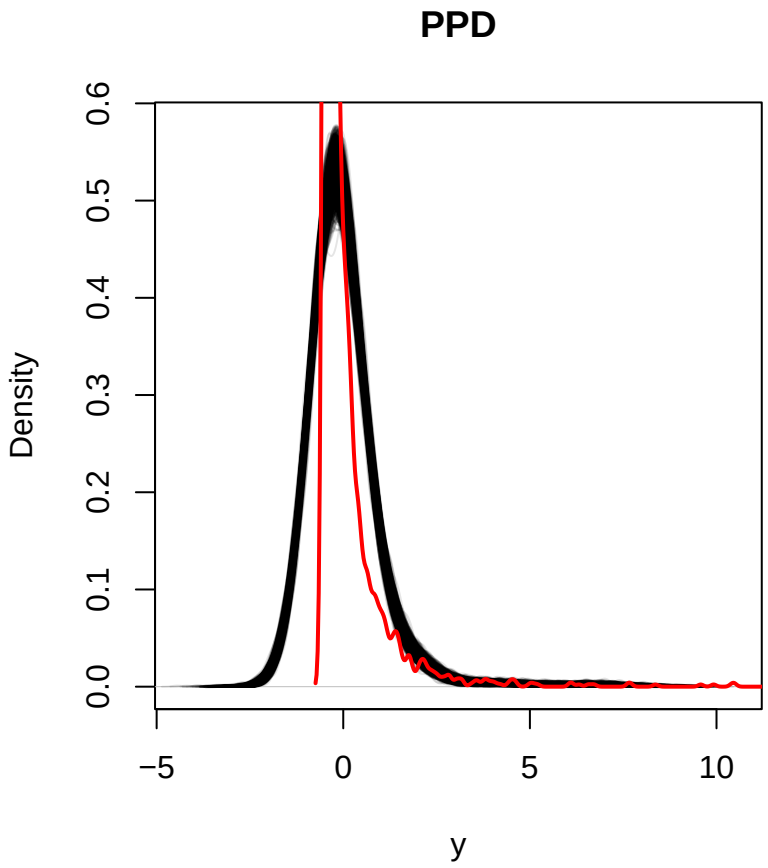
```
alpha <- sample_fit$alpha

data.frame(alpha) %>%
  pivot_longer(everything(), values_to = "individual") %>%
  mutate(ind = as.numeric(gsub("X", "", name))) %>%
  group_by(name) %>%
  summarise(low = quantile(individual, .025),
            upp = quantile(individual, .975),
            med = quantile(individual, .5)) %>%
  ggplot(aes(x = name)) +
  geom_point(aes(y = med), alpha = .5) +
  geom_errorbar(aes(ymin = low, ymax = upp), alpha = .5) +
  geom_hline(yintercept = 0, col = "red", lty = 2) +
  ggtitle("Individual Heterogeneity") +
  ylab(expression(alpha[i])) +
  xlab("Individual") +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.line = element_line(),
        title = element_text(size = 10))
```



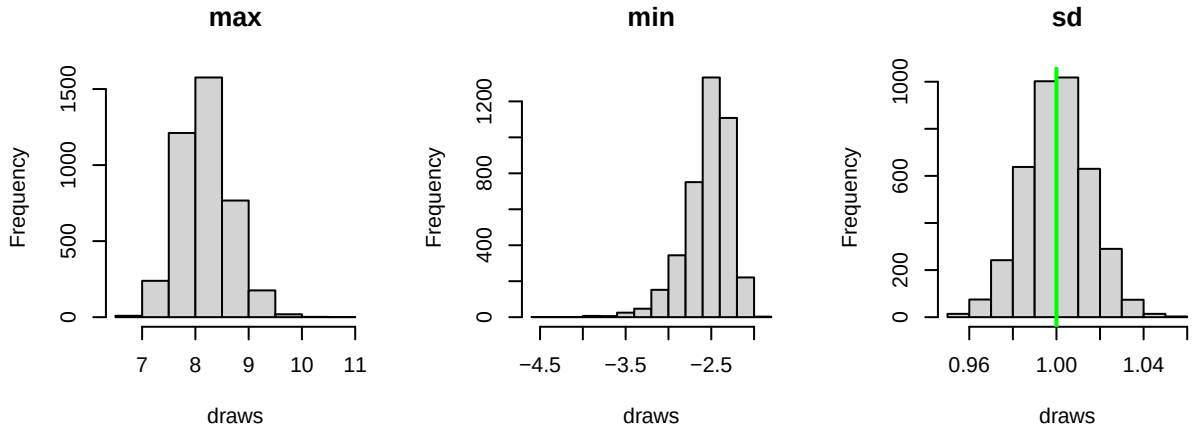
Model assessment: Posterior Predictive Checks (PoPC)

Checking model fit also follows the same procedure illustrated for the Normal linear regression case:

```
y_rep <- sample_fit$y_rep
plot_prior_dens(y_rep, std_earn, main = "PPD")
```




```
par(mfrow=c(1,3), pty="s")
hist(apply(y_rep, 1, max), main = "max", xlab = "draws")
abline(v=max(std_earn),col="red", lwd=2)
hist(apply(y_rep, 1, min), main = "min", xlab = "draws")
abline(v=min(std_earn),col="blue", lwd=2)
hist(apply(y_rep, 1, sd), main = "sd", xlab = "draws")
abline(v=sd(std_earn),col="green", lwd=2)
```



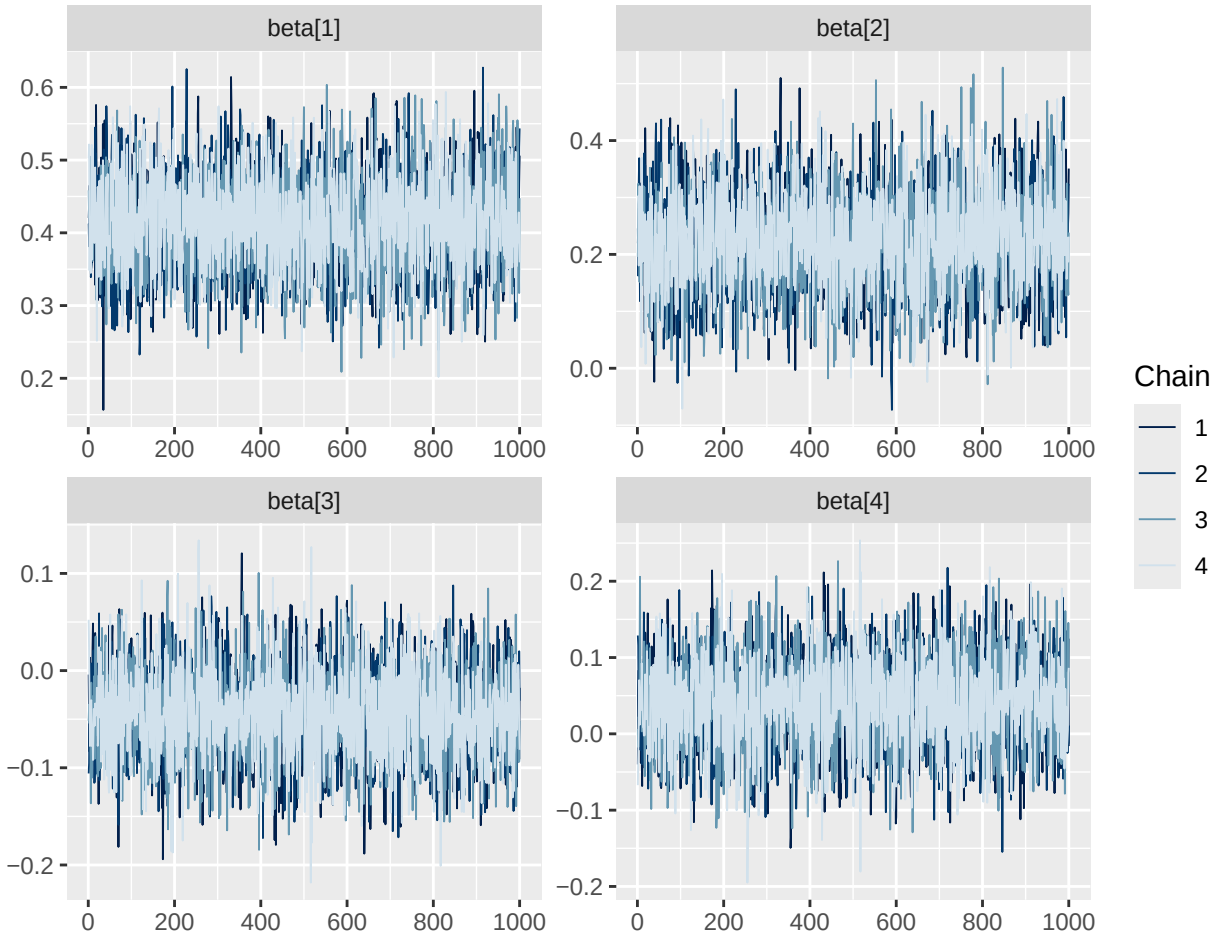

```
par(mfrow=c(1,1), pty="m")
```

As expected, both the density plot and the auxiliary `min`, `max` and `sd` distribution indicate insufficient model fit. This most likely results from the poor choice of likelihood for the outcome variable.

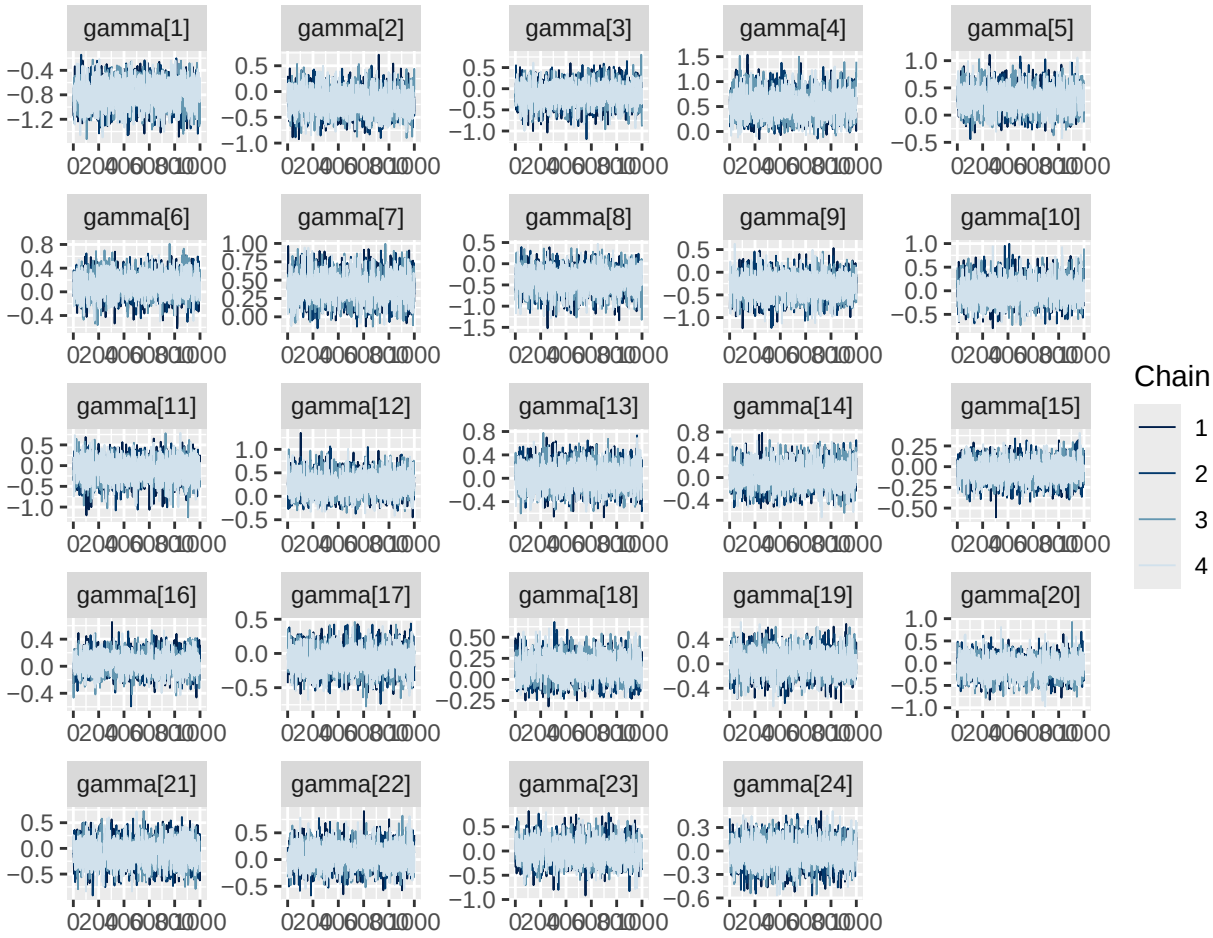
Convergence Diagnostics

Trace Plots

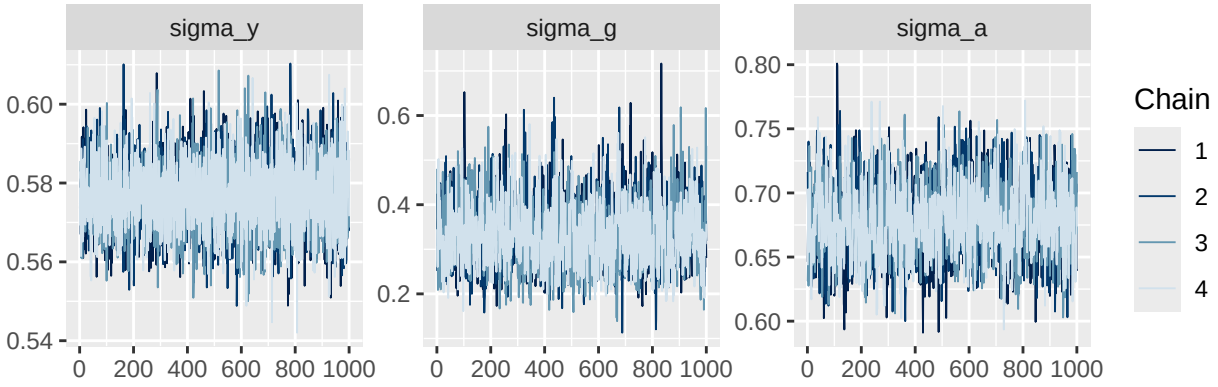
```
params_b <- paste0("beta[", 1:dat_list$K, "]")
params_g <- paste0("gamma[", 1:dat_list$L, "]")
params_s <- c("sigma_y", "sigma_g", "sigma_a")
mcmc_trace(fit, pars = params_b)
```




```
mcmc_trace(fit, pars = params_g)
```

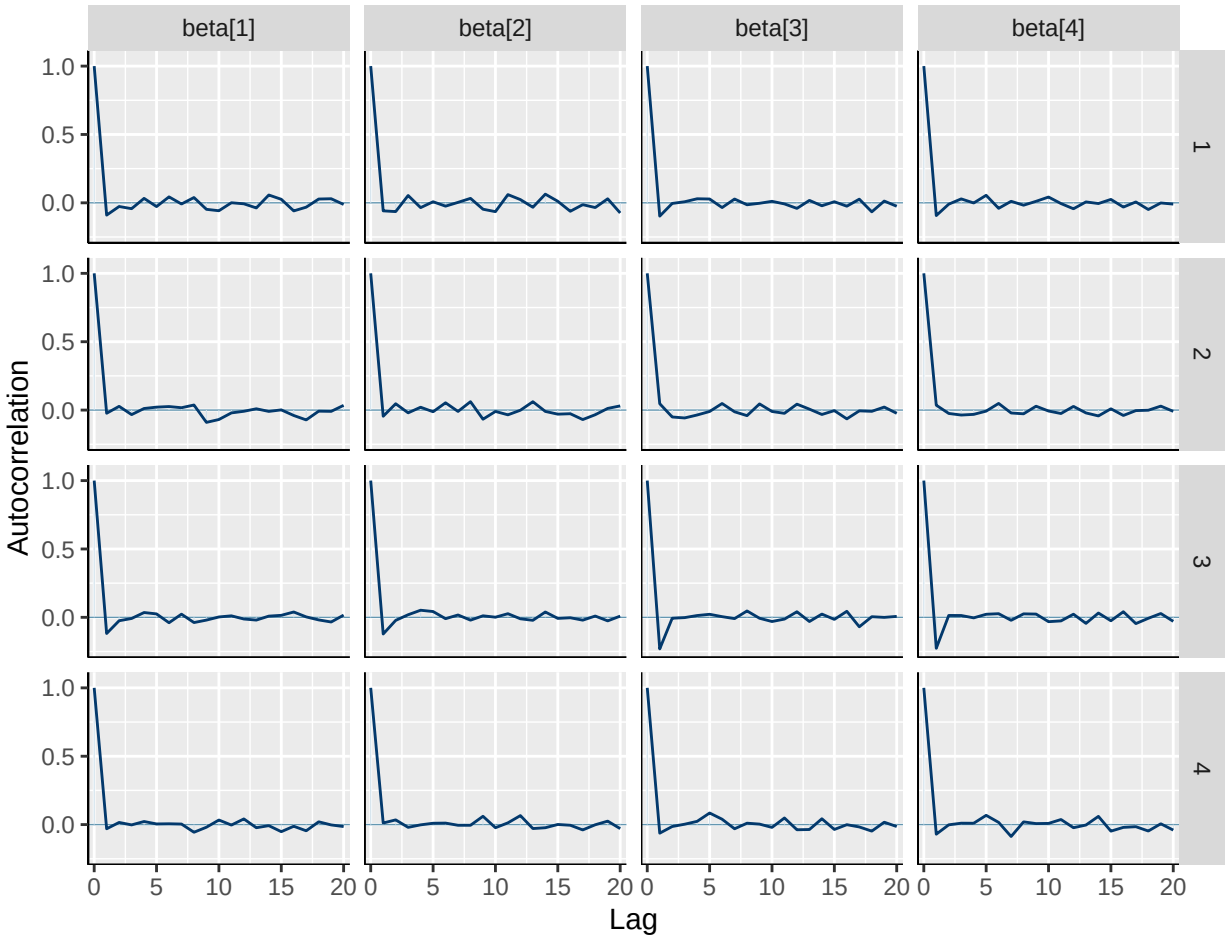



```
mcmc_trace(fit, pars = params_s) + theme(aspect.ratio = 1)
```

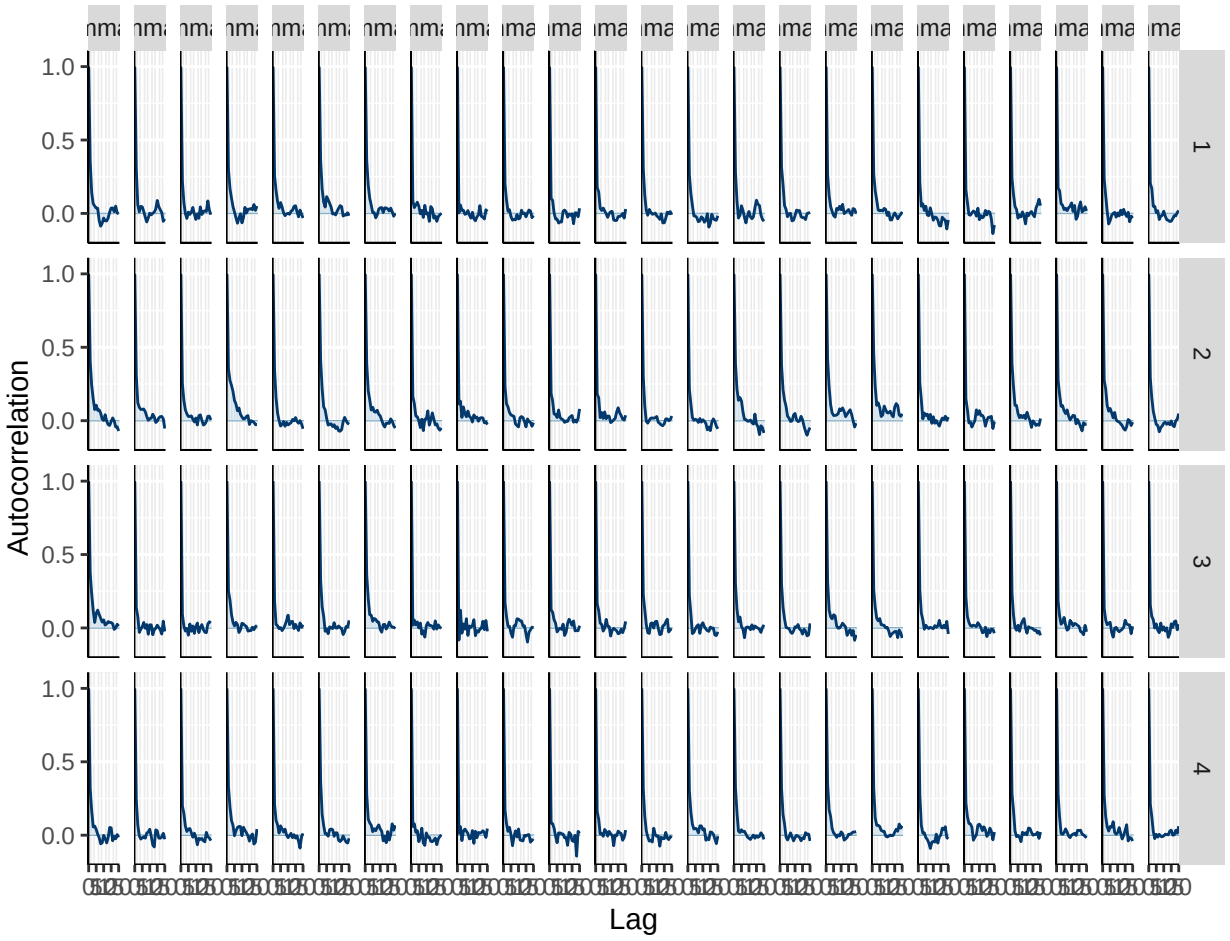


Autocorrelation Plots

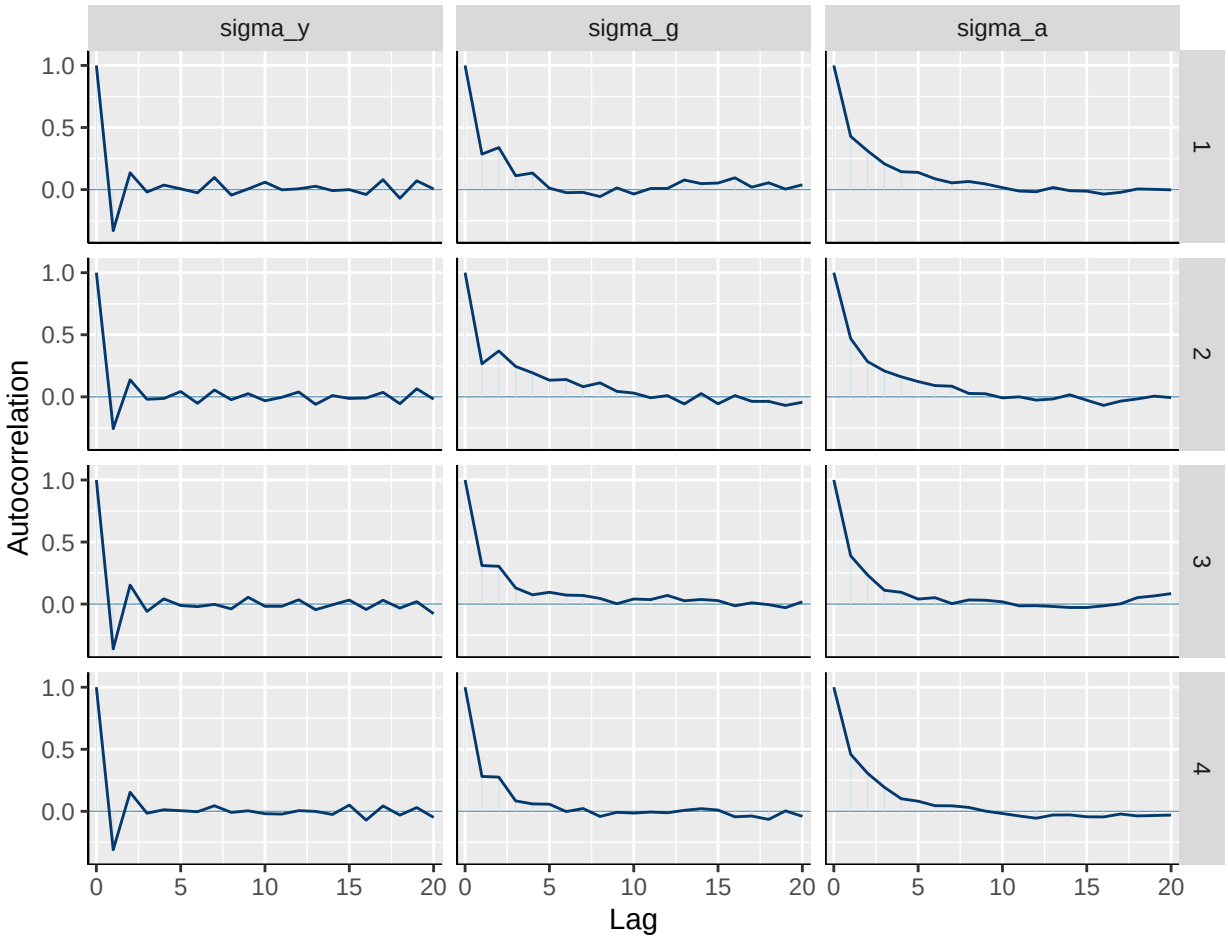
```
mcmc_acf(fit, pars = params_b)
```




```
mcmc_acf(fit, pars = params_g)
```

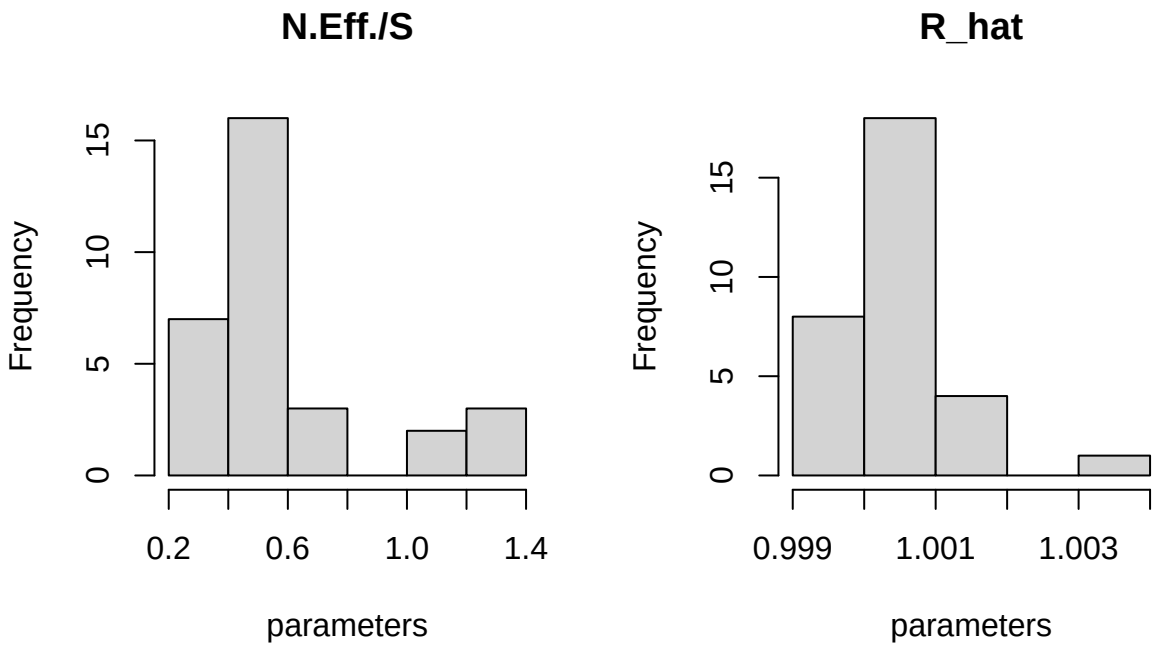



```
mcmc_acf(fit, pars = params_s)
```



Effective Sample Size and R-measure

```
par(mfrow=c(1,2), pty="s")
hist(neff_ratio(fit, pars = c(params_b, params_g, params_s)),
     main = "N.Eff./S", xlab = "parameters")
hist(rhat(fit, pars = c(params_b, params_g, params_s)),
     main = "R_hat", xlab = "parameters")
```




```
par(mfrow=c(1,1), pty="m")
```

References

Chamberlain, G. (1982). Multivariate regression models for panel data. Journal of econometrics, 18(1), 5-46.

Mundlak, Y. (1978). On the pooling of time series and cross section data. Econometrica: journal of the Econometric Society, 69-85.