Introduction to Bayesian linear regression with brms

Stefano Coretta

18/01/2020

Road map

1. Basic concepts.
2. Choosing priors.
3. Model fitting with brms.
4. Normal.
5. Log-normal.
6. Binomial (Bernoulli).
7. Bayesian inference.

Learning outcomes

- Understand the basic concepts behind Bayesian statistics and how it differs from frequentist statistics.
- Understand priors and how to choose them.
- Be able to set up and fit a Bayesian linear regression model using brms.
- Understand Bayesian inference methods.

BASIC CONCEPTS

Random variables

- We have a question about the world, so we collect data (sample from a population).
    - y = $(y_1, y_2, y_3, y_4, ..., y_n)$
- We want to know how the data (the sample y) was generated.
- In probability theory, data is generated by a random variable Y.

Random variables

- Y is a variable whose value is generated by a random event.
- Y is uncertain.
    - We can describe Y as a probability distribution.

Random variables

- Probability distribution.
  - A list of the values a random variable could take on along with their corresponding probability.
- Probability distributions can be expressed by a set of parameters $\Theta = (\theta_1, ..., \theta_n)$.
- Some probability distributions:
  - $Normal(\mu, \sigma)$,
  - $Binomial(n, p)$,
  - $Poisson(\lambda)$

Continuous random variables

$$y_i \sim Normal(\mu, \sigma)$$

Frequentist vs Bayesian view

- Parameters: μ, σ, p, λ, ...
- Frequentist view:
  - The parameters are **fixed** (they are unknown but certain).
  - They take on a specific value.
- Bayesian view:
  - The parameters are **random variables** (they are unkown and uncertain).
  - We describe each parameter as a probability distribution, expressed by a set of **hyperparameters**.

Priors

- We want to estimate the parameters μ and σ from the data.
- We can incorporate previous knowledge (belief) about the parameters using **priors** (*prior probability distributions*).
    - We specify the hyperparameters of the prior probability distributions.
- Priors are chosen based on expert knowledge, previous studies, pilot data...
    - Priors must **not** be chosen based on inspection of the data to be analysed.

Bayesian belief update

$$observed\ data \times prior\ belief = posterior\ belief$$

Bayesian belief update

https://nanx.shinyapps.io/conjugate-normal-umkv/

CHOOSING PRIORS

Italian F1

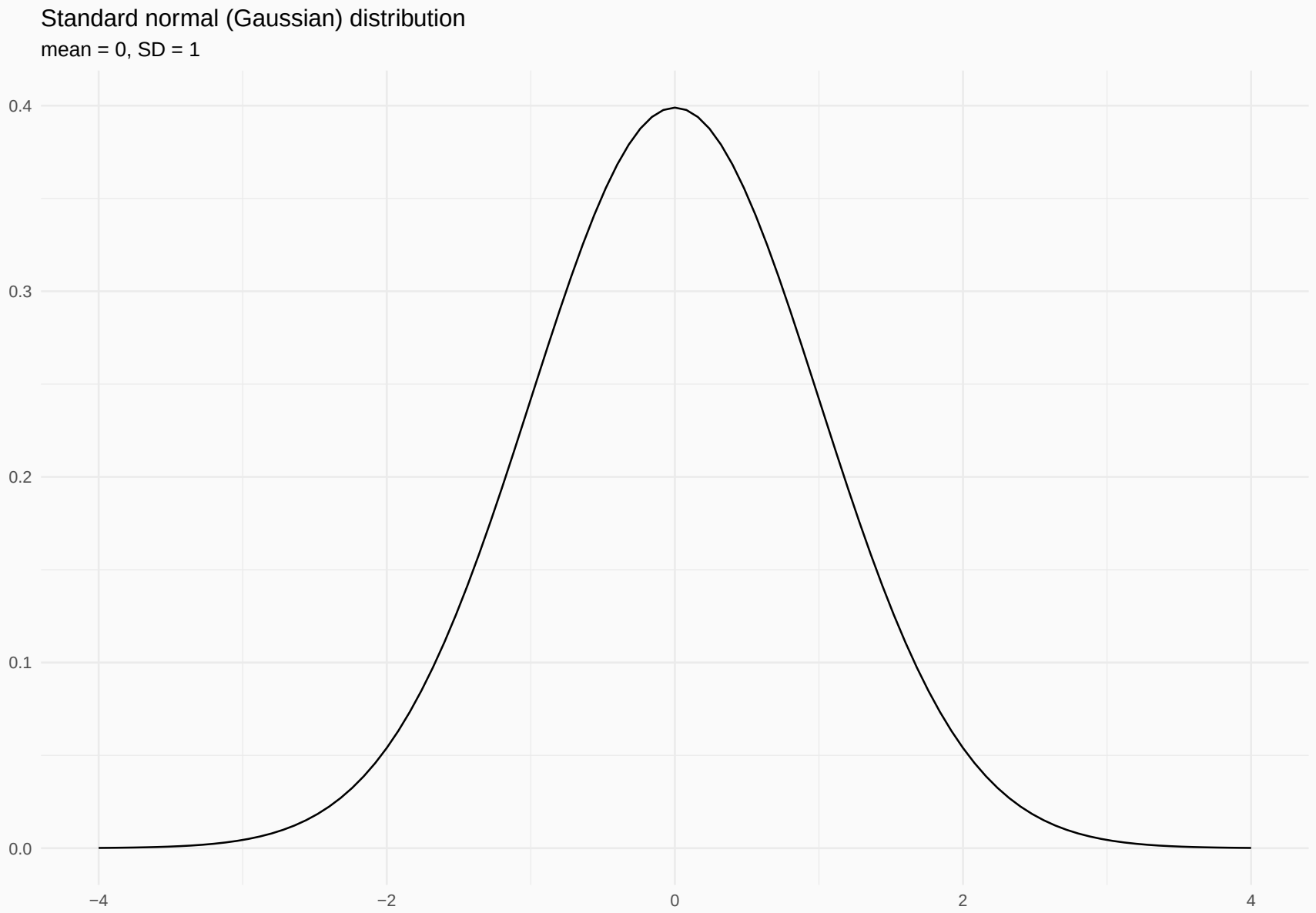
- Toy example with F1.
- Data:
  - Italian.
  - 7 speakers.
  - CVCV words in frame sentence.
    - C1 = /p/.
    - V1 = V2 = /a, o, u/.
    - C2 = /t, d, k, g/.

Italian F1

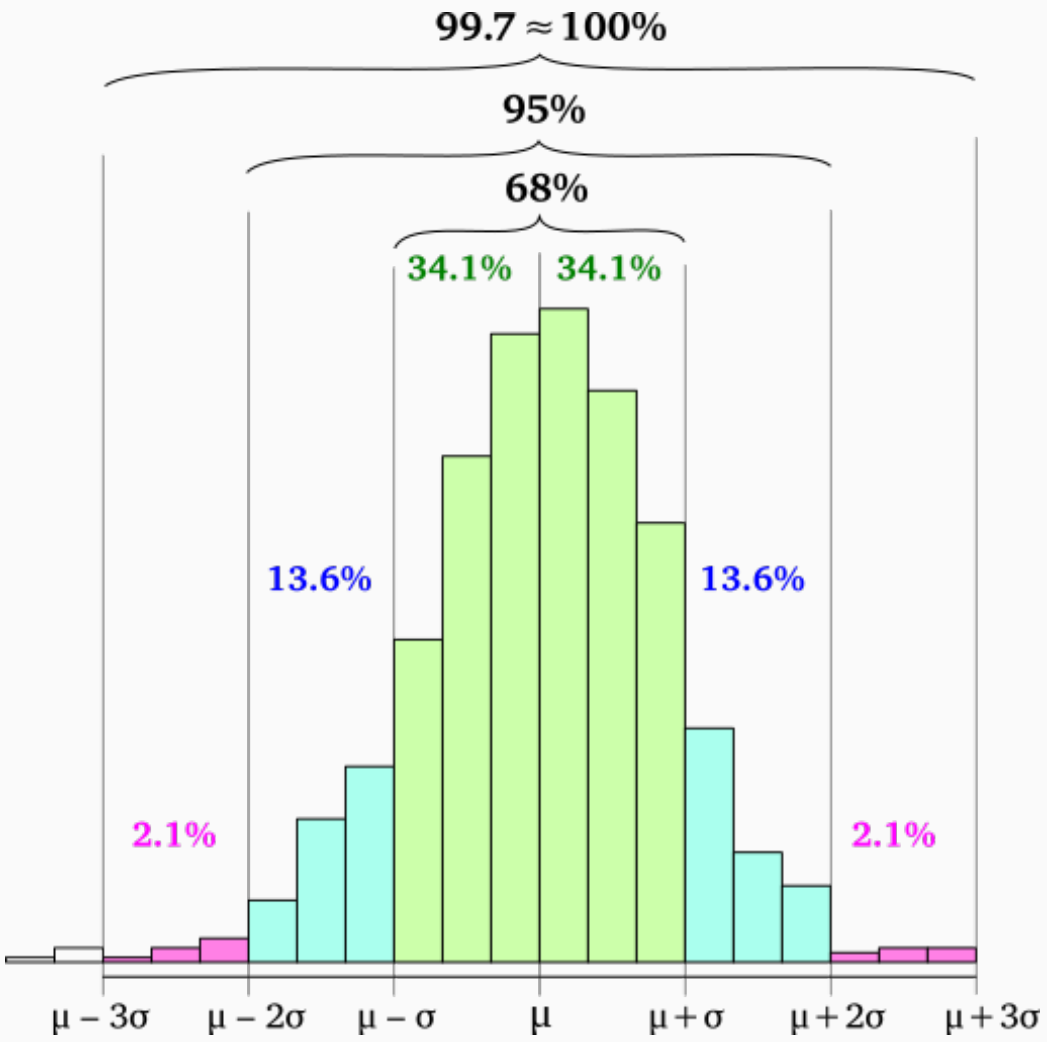
Formant values are roughly distributed according to a normal (Gaussian) distribution.

$$F1_i \sim Normal(\mu, \sigma)$$

Normal (Gaussian) distribution



Normal (Gaussian) distribution



Melikamp, https://commons.wikimedia.org/wiki/File:Empirical_rule_histogram.svg (CC BY-SA 4.0)

Italian F1

- $Normal(\mu, \sigma)$ has two parameters, μ and σ.
- μ and σ are random variables (unkown and uncertain).
- We express these parameters as priors (probability distributions with hyperparameters).

Priors

- Informative and weakly informative priors.
- Uninformative or diffuse priors.
    - Uniform distribution.
- Regularising priors.

$$vot_i \sim Normal(\mu, \sigma)$$

What prior for μ?

Italian F1

$$vot_i \sim Normal(\mu, \sigma)$$

$$\mu \sim Normal(\mu_1, \sigma_1)$$

Italian F1

```
ggplot() +
  aes(x = c(-2000, 2000)) +
  stat_function(fun = dnorm, n = 101, args = list(0, 500)) +
  labs(
    title = expression(paste("Prior for ", mu)),
    subtitle = expression(paste(mu, " = 0,  ", sigma, " = 500")),
    x = element_blank(),
    y = element_blank()
  )
```

Italian F1

```
ggplot() +
  aes(x = c(-2000, 2000)) +
  stat_function(fun = dnorm, n = 101, args = list(0, 500
  labs(
    title = expression(paste("Prior for ", mu)),
    subtitle = expression(paste(mu, " = 0,  ", sigma, "
    x = element_blank(),
    y = element_blank()
  )
```

Italian F1

- $Normal(0, 500)$ as the prior for μ.
- This means that we believe μ to be between -1000 and +1000 at 95% confidence.
    - 95% credible interval (CI) = $[\mu_1 + 2\sigma_1, \mu_1 - 2\sigma_1]$.

Italian F1

$$vot_i \sim Normal(\mu, \sigma)$$

$$\mu \sim Normal(0, 500)$$

What about σ?

Italian VOT

$$vot_i \sim Normal(\mu, \sigma)$$

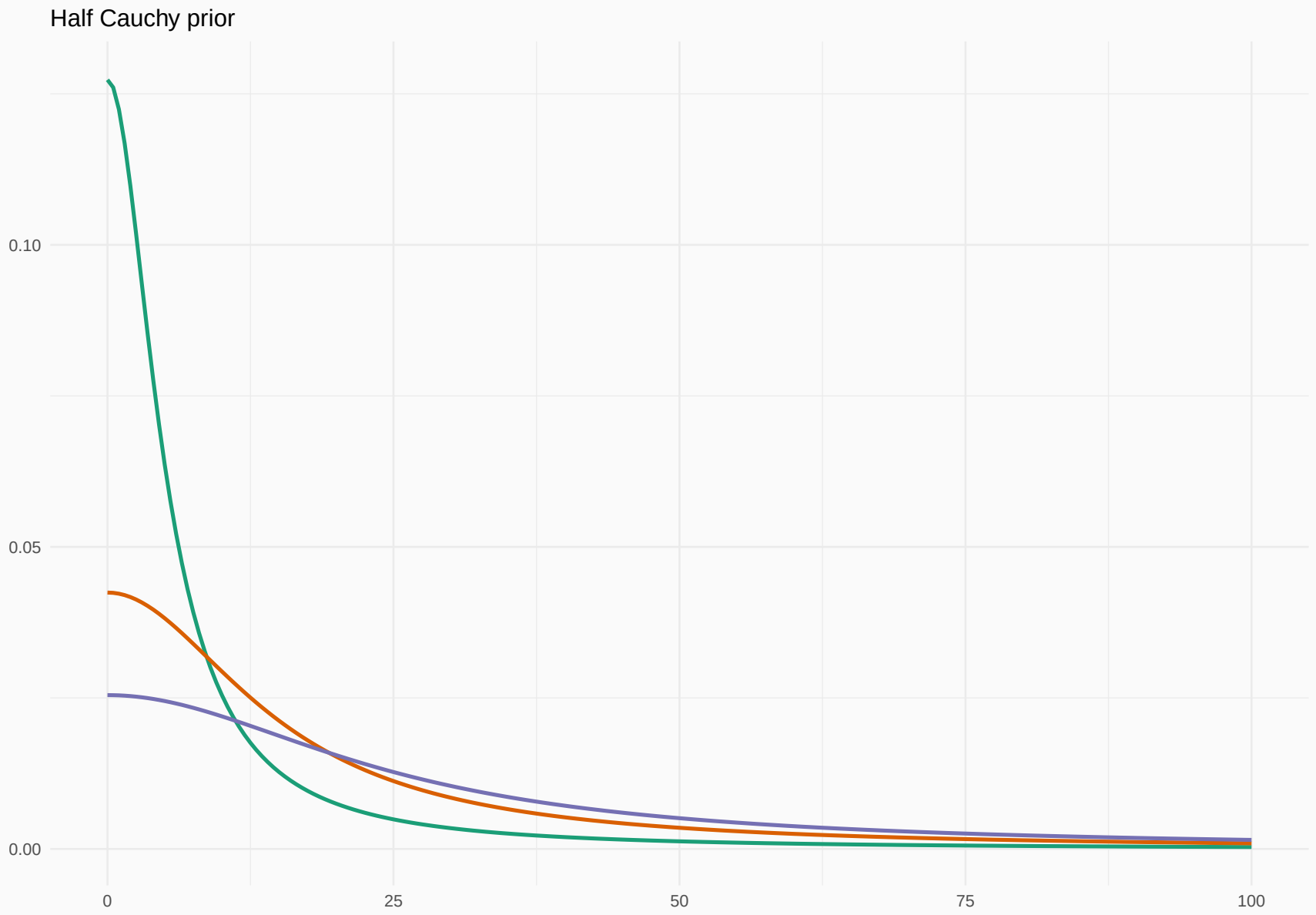
$$\mu \sim Normal(0, 500)$$

$$\sigma \sim HalfCauchy(x_0, \gamma)$$

Cauchy distribution

```
ggplot() +
  aes(x = c(-100, 100)) +
  stat_function(fun = dcauchy, n = 201, args = list(0, 5), colour = "#1b9e77", size = 1) +
  stat_function(fun = dcauchy, n = 201, args = list(0, 15), colour = "#d95f02", size = 1) +
  stat_function(fun = dcauchy, n = 201, args = list(0, 25), colour = "#7570b3", size = 1) +
  labs(
    title = "Cauchy distribution",
    x = element_blank(),
    y = element_blank()
  )
```

Half Cauchy prior



Half cauchy prior

```
library(HDInterval)
inverseCDF(c(0.025, 0.975), phcauchy, 5)
```

```
## [1]   0.1964676 127.2584987
```

```
inverseCDF(c(0.025, 0.975), phcauchy, 15)
```

```
## [1]   0.5893621 381.7754937
```

```
inverseCDF(c(0.025, 0.975), phcauchy, 25)
```

```
## [1]   0.9822792 636.2924897
```

Italian F1

$$vot_i \sim Normal(\mu, \sigma)$$

$$\mu \sim Normal(0, 500)$$

$$\sigma \sim HalfCauchy(0, 25)$$

MODEL FITTING

Italian F1

- We have a model which incorporates our (vague) knowledge about F1 (through the priors for μ and σ).
- Now we want to obtain the **posterior distributions** of μ and σ.
    - The posterior distribution is the prior distribution *conditioned* on the data.
- **brms** R package: Bayesian Regression Models using Stan (Bürkner, 2018).

Installation

Safe method:

- **Install Rstan first**: https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started (see *Installation of Rstan*, *Checking the C++ Toolchain*, and *Configuration of the C++ Toolchain*).
    - Note that details in *Checking the C++ Toolchain* differ depending on your OS.
- **Install brms**: https://github.com/paul-buerkner/brms#how-do-i-install-brms.

brms

- Stan (Stan Development Team, 2017).
    - Statistical programming language written in C++.
    - Fit Bayesian models (calculate posterior distributions).
- Calculation can be complex and/or impossible, so we take many samples from the data and from the possible parameter values to find the posterior distributions of the hyperparameters.
    - Markov Chain Monte Carlo (MCMC) sampling using the No-U-Turn sampler (NUTS).

brms

- brms translates R code into Stan code.
- Stan code is run in Stan via Rstan, an R interface to Stan.
- `brm()` function from brms.
  - lme4 syntax (`y ~ x + (1|w)`).
  - Creates a Stan model, which is compiled (in C++) and run.

brms

```
library(brms)

vot1 <- brm(
  <model_formula>,
  <family>,
  <prior>,
  <data>,
  chains = 4,
  iter = 2000
)
```

brms

```
library(brms)

vot1 <- brm(
  vot ~ 1,
  family = gaussian(),
  <prior>,
  data = ita_egg,
  chains = 4,
  iter = 2000
)
```

Get prior

```
get_prior(
  f1 ~ 1,
  family = gaussian(),
  data = f_end
)
```

```
##                   prior     class coef group resp dpar nlpar bound
## 1 student_t(3, 495, 167) Intercept
## 2   student_t(3, 0, 167)     sigma
```

Prior predictive checks

```r
nsim <- 1000
nobs <- 1000

y <- matrix(rep(NA, nsim * nobs), ncol = nobs)

mu <- rnorm(nsim, 0, 500)
sigma <- rhcauchy(nsim, 25)

for (i in 1:nsim) {
  y[i,] <- rnorm(nobs, mean = mu[i], sd = sigma[i])
}
```

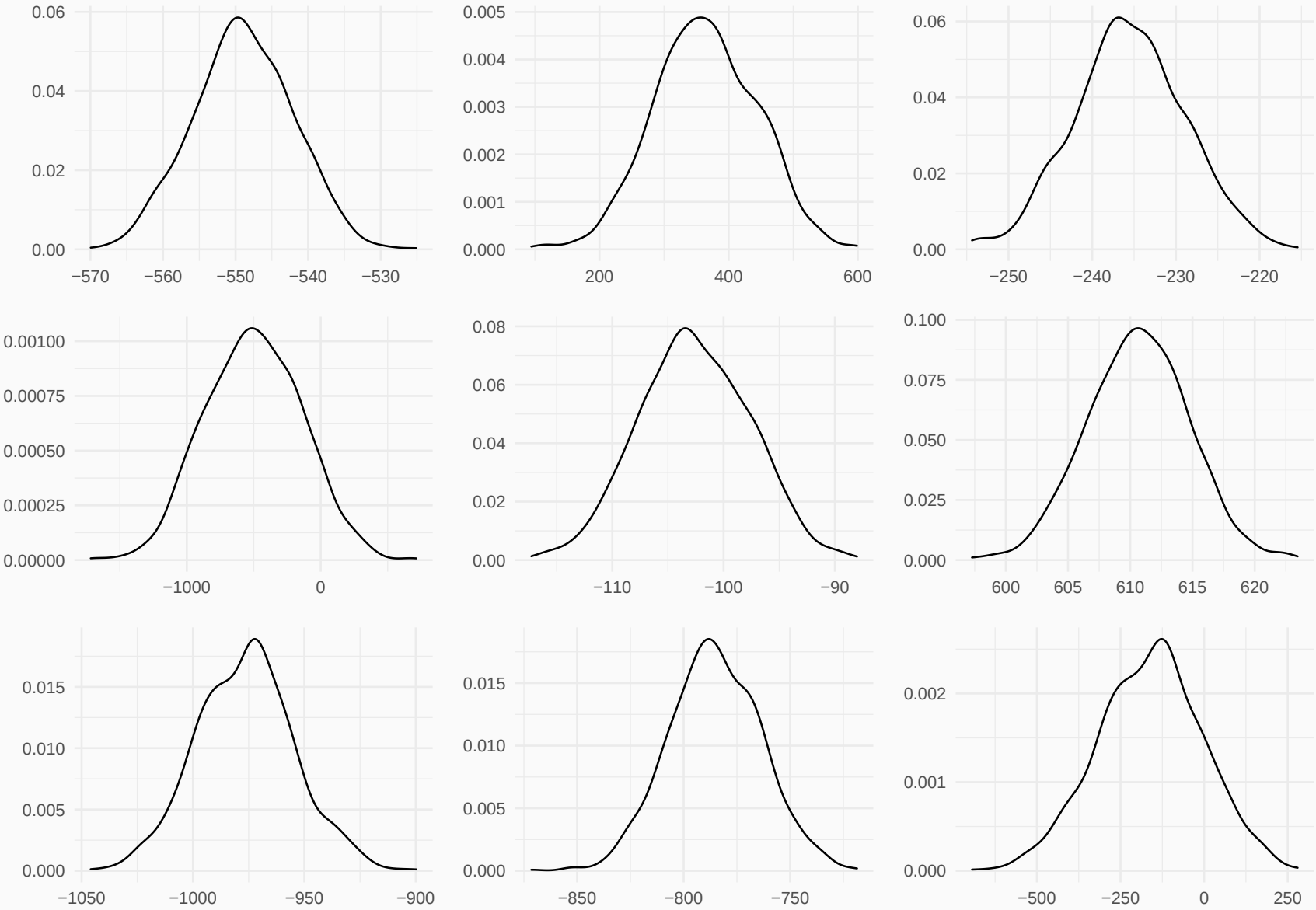
Prior predictive checks

```
rand_sample <- sample(1:nsim, 9, replace = FALSE)
plots <- list()

j = 1

for (i in rand_sample) {
  my_data <- enframe(y[i,], name = NULL)
  plots[[j]] <- ggplot(data = my_data) +
    aes(x = value) +
    geom_density() +
    labs(x = element_blank(), y = element_blank())
  j = j + 1
}
```

Prior predictive checks



Set prior

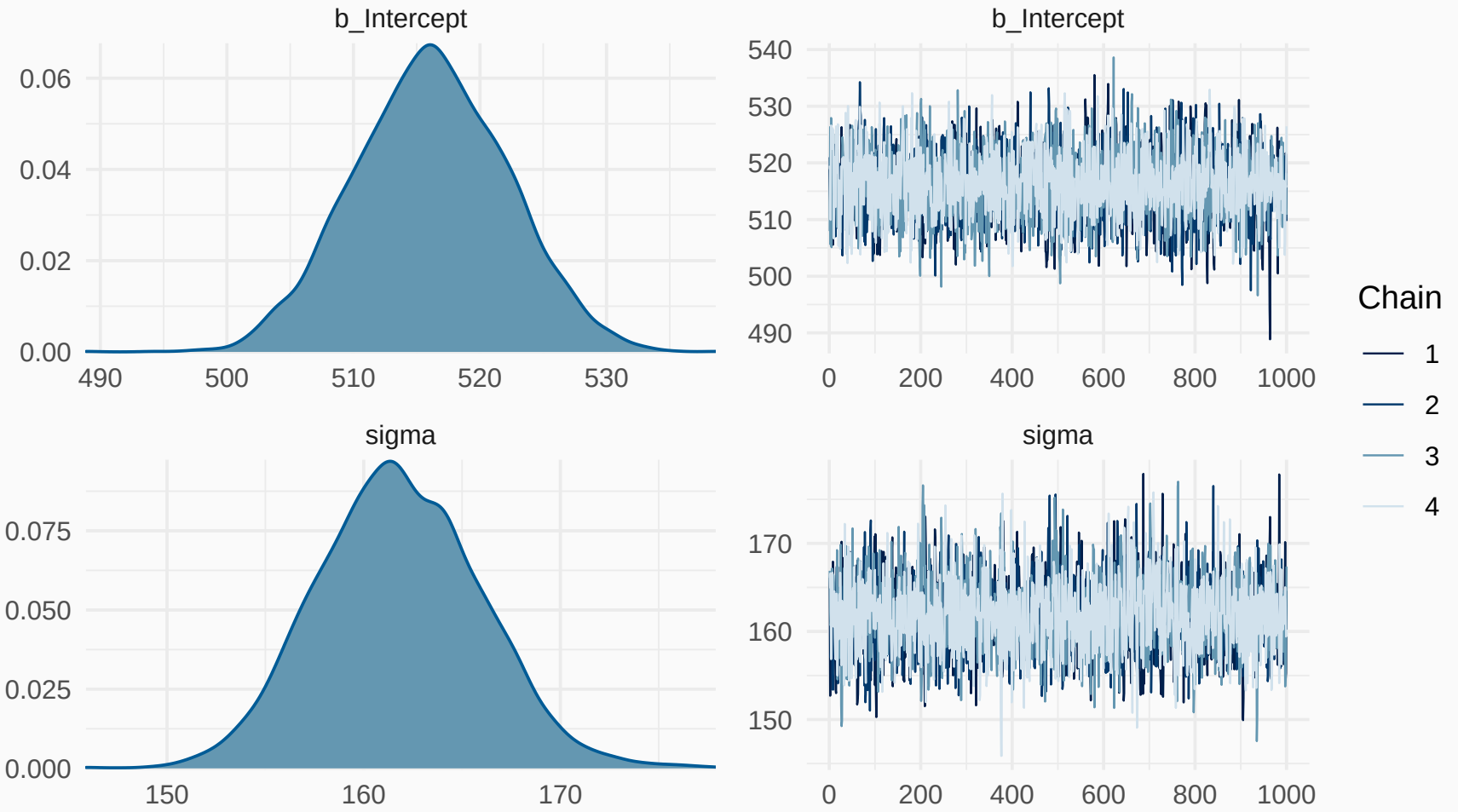
```
priors <- c(
  prior(normal(0, 500), class = Intercept),
  prior(cauchy(0, 25), class = sigma)
)
```

Run the model

```
f1_bm <- brm(
  f1 ~ 1,
  family = gaussian(),
  prior = priors,
  data = f_end,
  chains = 4,
  iter = 2000,
  file = "./cache/f1"
)
```

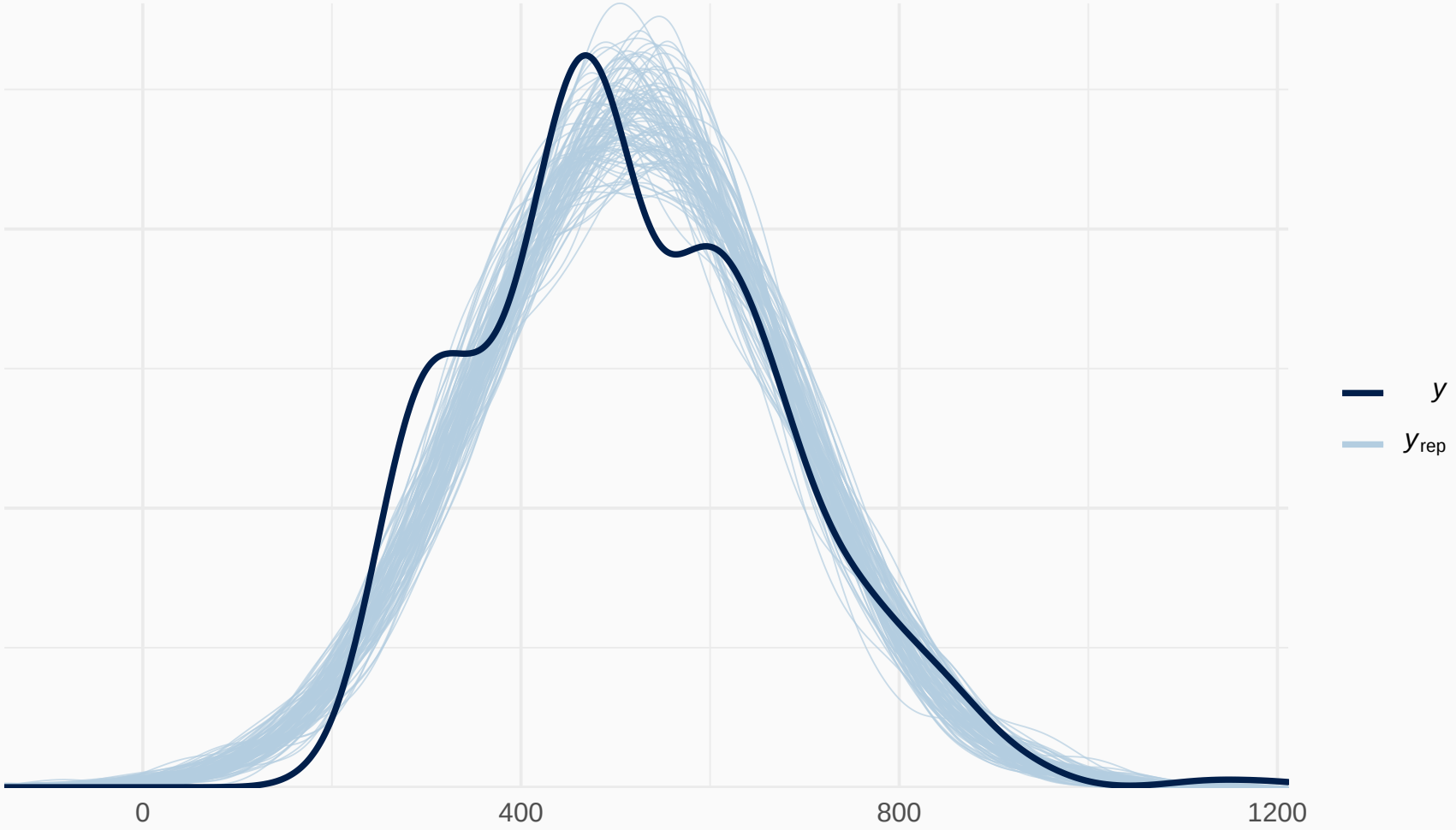
Plot model

```
plot(f1_bm, ask = FALSE)
```



Posterior predictive check

```
pp_check(f1_bm, nsamples = 100)
```



Model summary

```
f1_bm
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: f1 ~ 1
##    Data: f_end (Number of observations: 748)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept   516.14      6.01   504.35   527.81 1.00     3358     2498
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma   161.81      4.18   153.95   170.03 1.00     3071     2682
##
## Samples were drawn 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).
```

Adding predictors

$$vot_i \sim Normal(\mu_i, \sigma)$$

$$\mu_i = \alpha + \beta_1 \times O_i + \beta_2 \times U_i$$

$$\alpha \sim Normal(\mu_1, \sigma_1)$$

$$\beta_1 \sim Normal(\mu_2, \sigma_2)$$

$$\beta_2 \sim Normal(\mu_3, \sigma_3)$$

$$\sigma \sim HalfCauchy(x_0, \gamma)$$

Adding predictors

$$vot_i \sim Normal(\mu_i, \sigma)$$

$$\mu_i = \alpha + \beta_1 \times O_i + \beta_2 \times U_i$$

$$\alpha \sim Normal(0, 500)$$

$$\beta_1 \sim Normal(0, 750)$$

$$\beta_2 \sim Normal(0, 750)$$

$$\sigma \sim HalfCauchy(0, 25)$$

Adding predictors

```
get_prior(
  f1 ~ 1 + vowel,
  family = gaussian(),
  data = f_end
)
```

```
##                     prior     class   coef group resp dpar nlpar bound
## 1                                 b
## 2                                 b vowelo
## 3                                 b vowelu
## 4 student_t(3, 495, 167) Intercept
## 5   student_t(3, 0, 167)     sigma
```

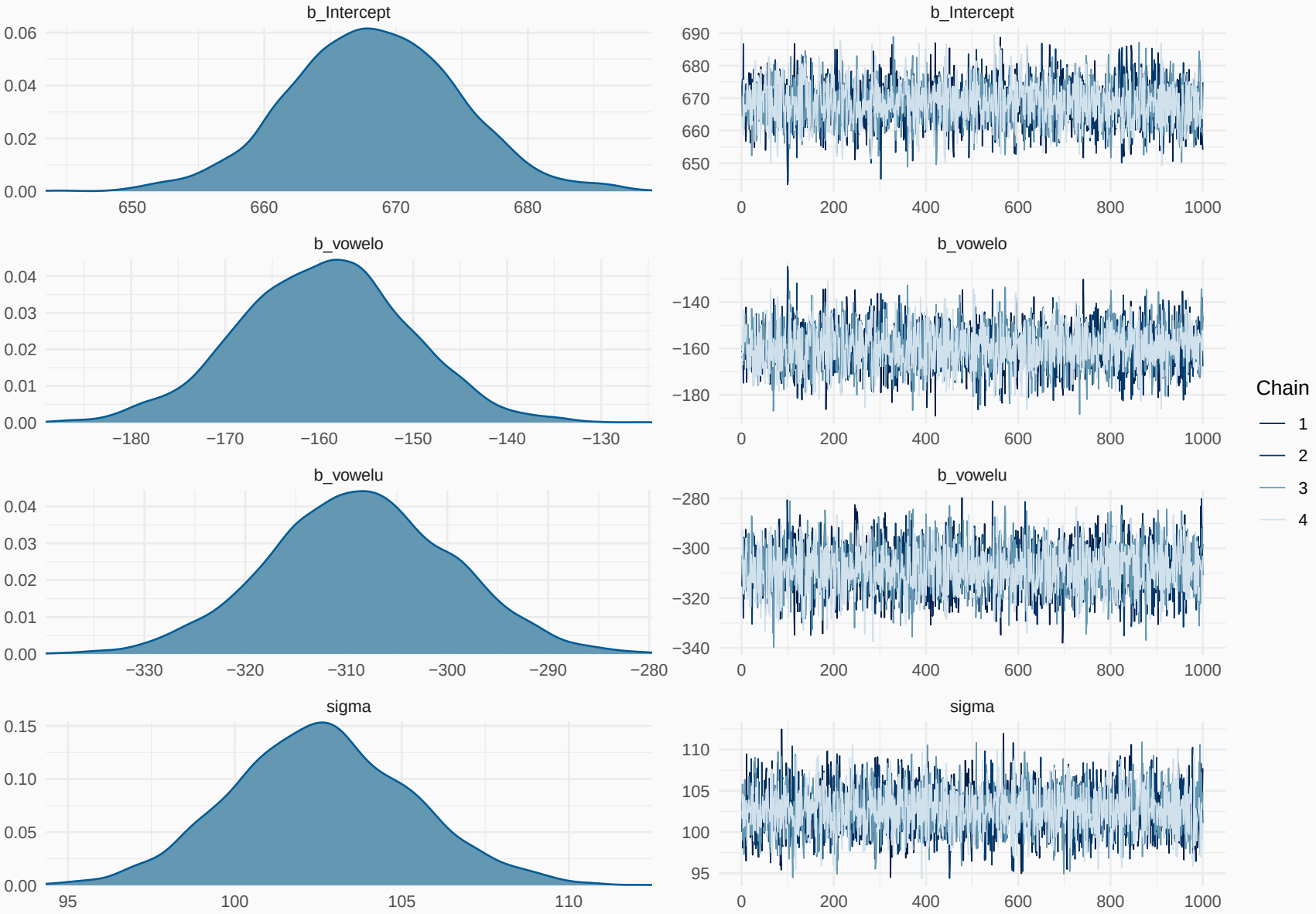
Adding predictors

```
priors <- c(
  prior(normal(0, 500), class = Intercept),
  prior(cauchy(0, 25), class = sigma),
  prior(normal(0, 750), class = b, coef = "vowelo"),
  prior(normal(0, 750), class = b, coef = "vowelu")
)
```

Adding predictors

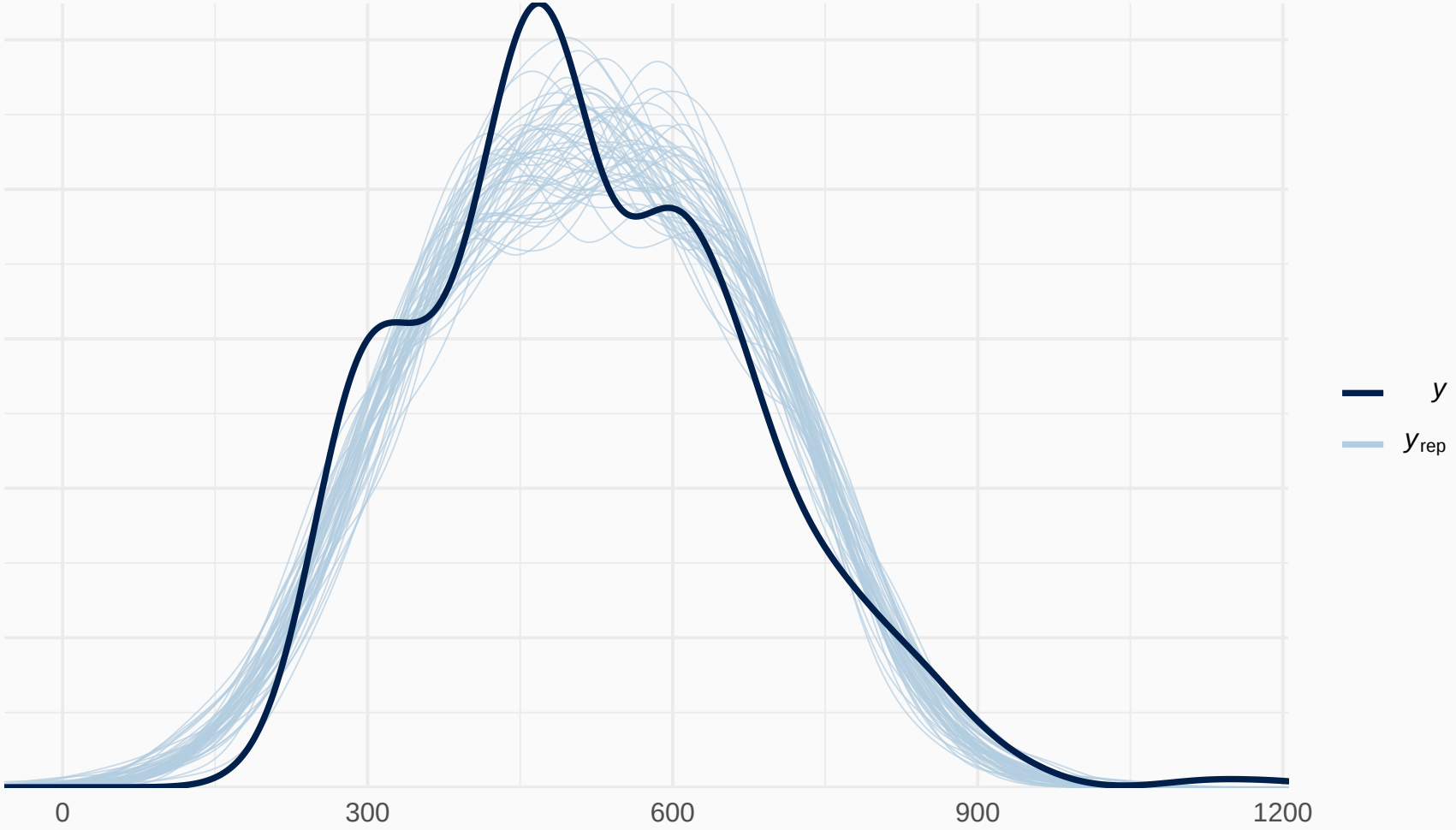
```
f1_2 <- brm(
  f1 ~ 1 + vowel,
  family = gaussian(),
  prior = priors,
  data = f_end,
  chains = 4,
  iter = 2000,
  file = "./cache/f1_2"
)
```

Adding predictors



Adding predictors

```
pp_check(f1_2, nsamples = 50)
```

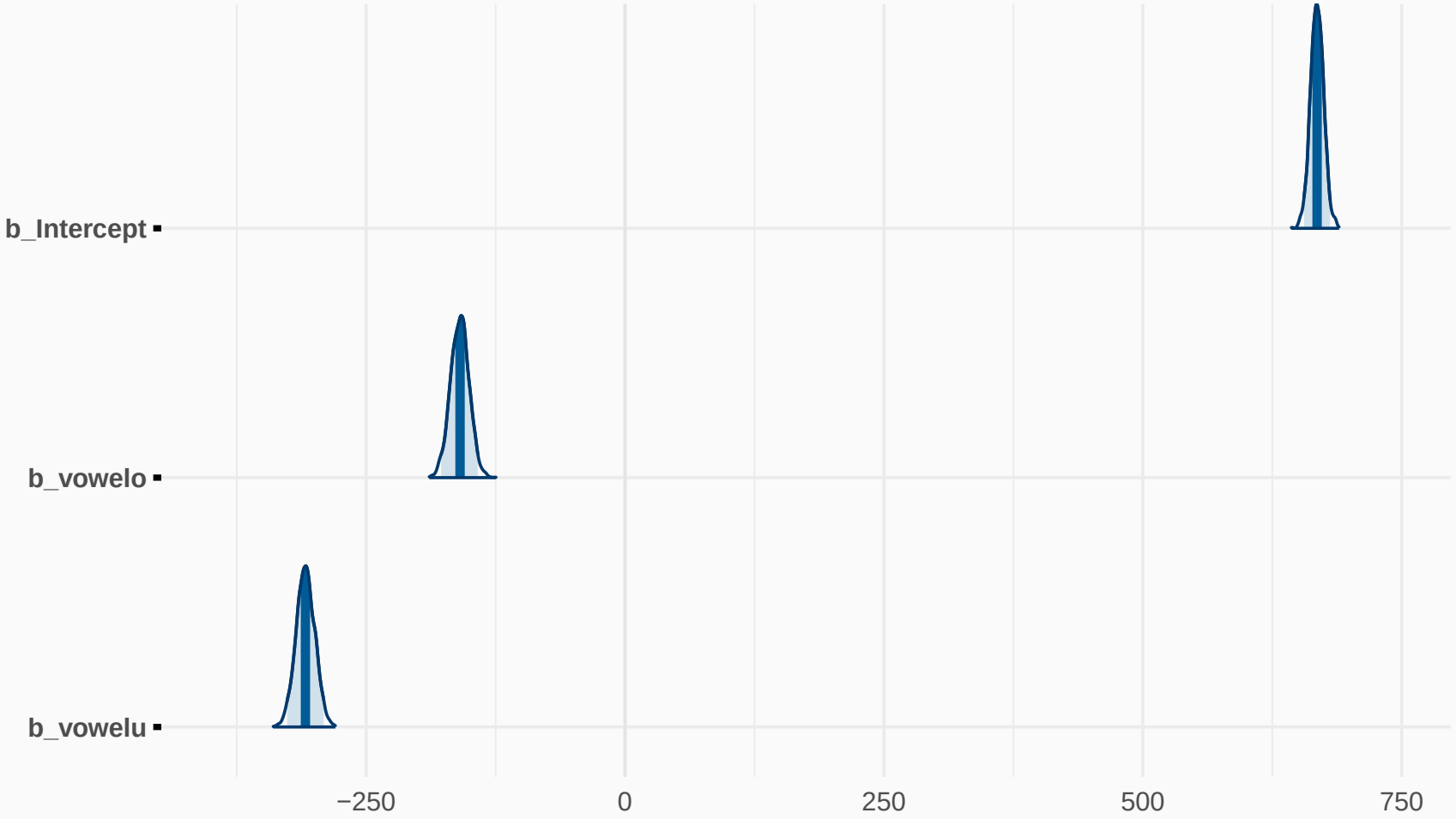


Adding predictors

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: f1 ~ 1 + vowel
##    Data: f_end (Number of observations: 748)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept   668.30      6.38   655.69   680.85 1.00     3249     2620
## vowelo     -159.54      8.94  -177.58  -142.24 1.00     3355     2903
## vowelu     -308.57      9.06  -326.33  -291.04 1.00     3281     2734
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma   102.60      2.70    97.38   108.10 1.00     3470     2818
##
## Samples were drawn 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).
```

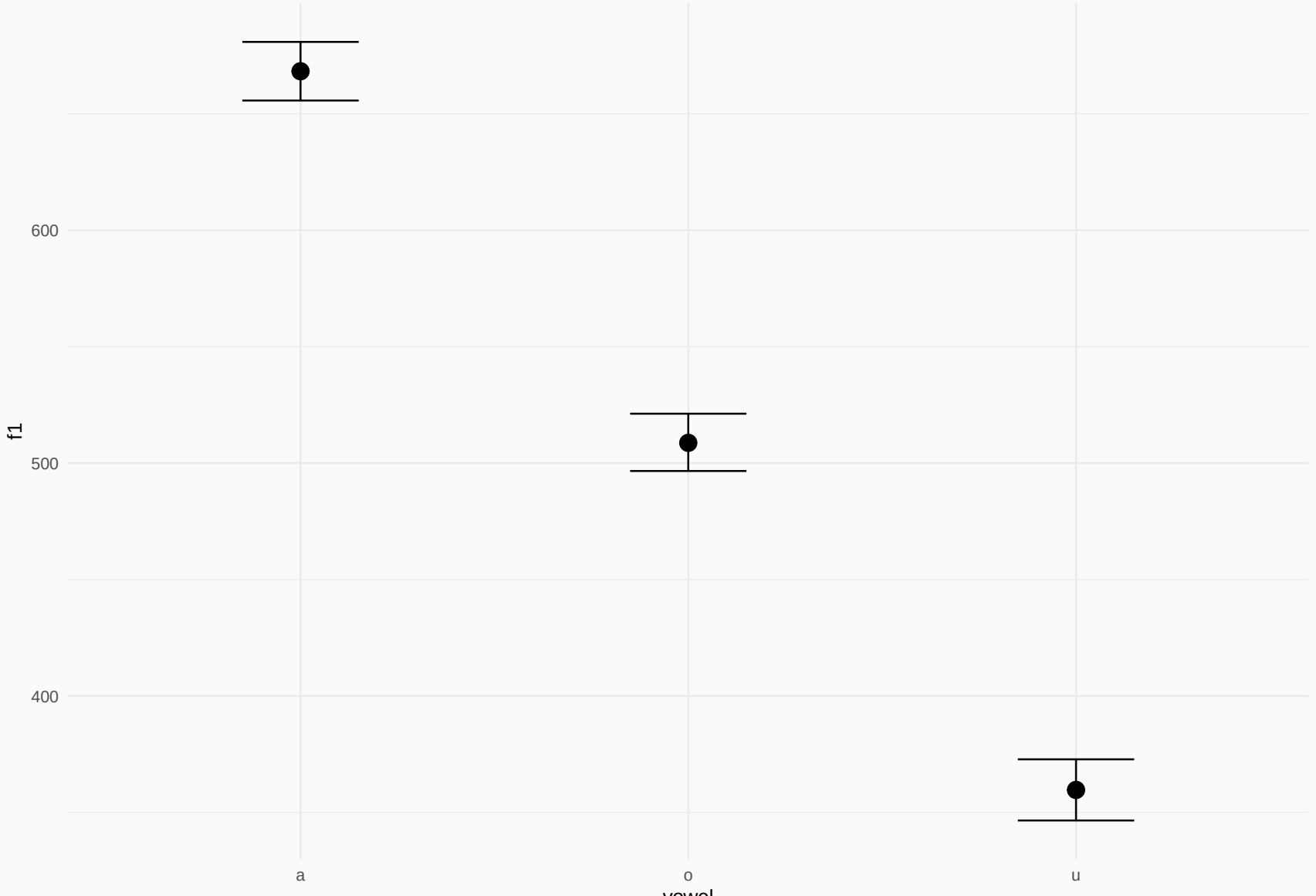
Adding predictors

```
mcmc_areas(f1_2, regex_pars = "b_", prob = 0.95)
```



Adding predictors

```
conditional_effects(f1_2)
```



References

Bürkner, Paul-Christian. 2018. Advanced Bayesian multilevel modeling with the R package brms. *The R Journal* 10(1). 395–411.

Stan Development Team. 2017. Stan: A C++ library for probability and sampling, version 2.14.0. http://mc-stan.org/.