Computational Bayesian data analysis

LOT winter school 2020

Bruno Nicenboim / Shravan Vasishth

2020-01-15

Bayesian Regression Models using ‘Stan’: brms

Examples 1: A single participant pressing a button repeatedly (A simple linear model)

Prior predictive distributions

The influence of priors: sensitivity analysis

Posterior predictive distributions

Comparing different likelihoods: The log-normal likelihood

- Deriving the posterior distribution analytically is possible for only a very limited number of cases.
- The denominator, the marginal likelihood, requires us to integrate the numerator:

$$p(\Theta|y) = \frac{p(y|\Theta) \cdot p(\Theta)}{\int_{\Theta} p(y|\Theta) \cdot p(\Theta) d\Theta} \tag{1}$$

Alternative: Deriving the posterior through sampling

We want to derive the posterior distribution of the Cloze probability of *"umbrella"*, θ:

- Data: a word (e.g., *"umbrella"*) was answered 80 out of 100 times,
- Likelihood: a binomial distribution
- Prior for θ: $Beta(a = 4, b = 4)$

We sample from the posterior distribution of θ:

- We use a probabilistic programming language,
- given enough samples we will have a good approximation of the real posterior distribution,
- say we got 20000 samples from the posterior distribution of the Cloze probability, θ:

0.836, 0.768, 0.795, 0.757, 0.785, 0.785, 0.75, 0.752, 0.747, 0.748, 0.756, 0.815, 0.775, 0.832, 0.787, 0.755, 0.818, 0.796, 0.795, 0.752, ...

The approximation of the posterior looks quite similar to the real posterior.[1]

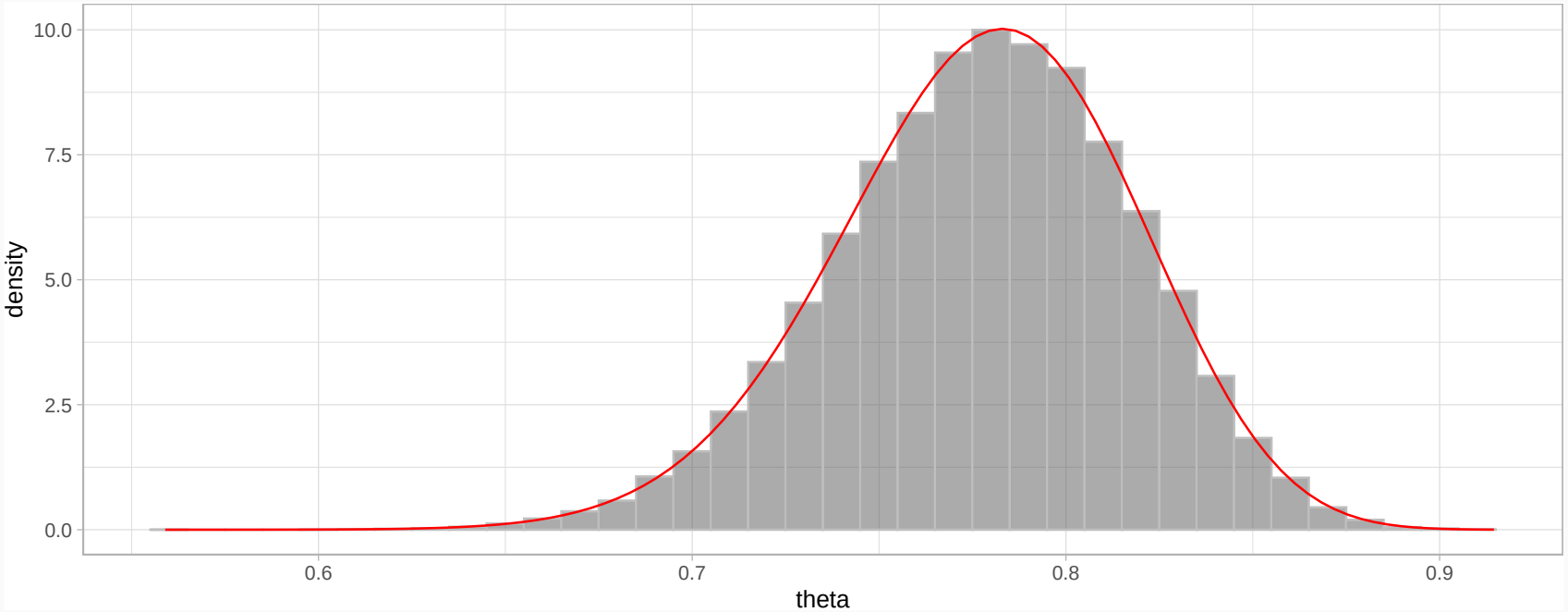


Figure 1: Histogram of the samples of θ from the posterior distribution calculated through sampling in gray; density plot of the exact posterior in red.

[1] The difference between the true and the approximated mean and variance are 0.00002 and -0.000003 respectively

Computational Bayesian data analysis:

Why now?

- increase in computing power
- appearance of probabilistic programming languages: WinBUGS (Lunn et al. 2000), JAGS (Plummer 2016), and more recently pymc3 (Salvatier, Wiecki, and Fonnesbeck 2016) and Stan (Carpenter et al. 2017).

Easier alternatives based on Stan:

- `rstanarm` (Goodrich et al. 2018)
- `brms` (Bürkner 2019)

Bayesian Regression Models using 'Stan': brms

Load the following:

```
set.seed(42)
library(MASS)
## be careful to load dplyr after MASS
library(dplyr)
library(tidyr)
library(purrr)
library(readr)
library(ggplot2)
library(brms)
## Save compiled models:
rstan_options(auto_write = TRUE)
## Parallelize the chains using all the cores:
options(mc.cores = parallel::detectCores())
library(bayesplot)
library(tictoc)
```

Examples 1: A single participant pressing a button repeatedly (A simple linear model)

We have data from a participant repeatedly pressing the space bar as fast as possible, without paying attention to any stimuli.

Data:
reaction times in milliseconds in each trial

Question:
How long does it take to press a key when there is no decision involved?

Assumptions:

1. There is a true underlying time, μ, that the participant needs to press the space bar.
2. There is some noise in this process.
3. The noise is normally distributed (this assumption is questionable given that reaction times are generally skewed; we fix this assumption later).

Formal model:

Likelihood for each observation n:

$$rt_n \sim Normal(\mu, \sigma) \tag{2}$$

(Bad) priors:

$$\begin{aligned} \mu &\sim Uniform(0, 60000) \\ \sigma &\sim Uniform(0, 2000) \end{aligned} \tag{3}$$

Fitting the model

We'll first load the data from `data/button_press.csv`:

```
df_noreading_data <-
  read_csv("./data/button_press.csv")
df_noreading_data
```

```
## # A tibble: 361 x 2
##       rt trialn
##    <dbl>  <dbl>
## 1    141      1
## 2    138      2
## 3    128      3
## 4    132      4
## 5    126      5
## # ... with 356 more rows
```

```
ggplot(df_noreading_data, aes(rt)) +
  geom_density() +
  ggtitle("Button-press data")
```

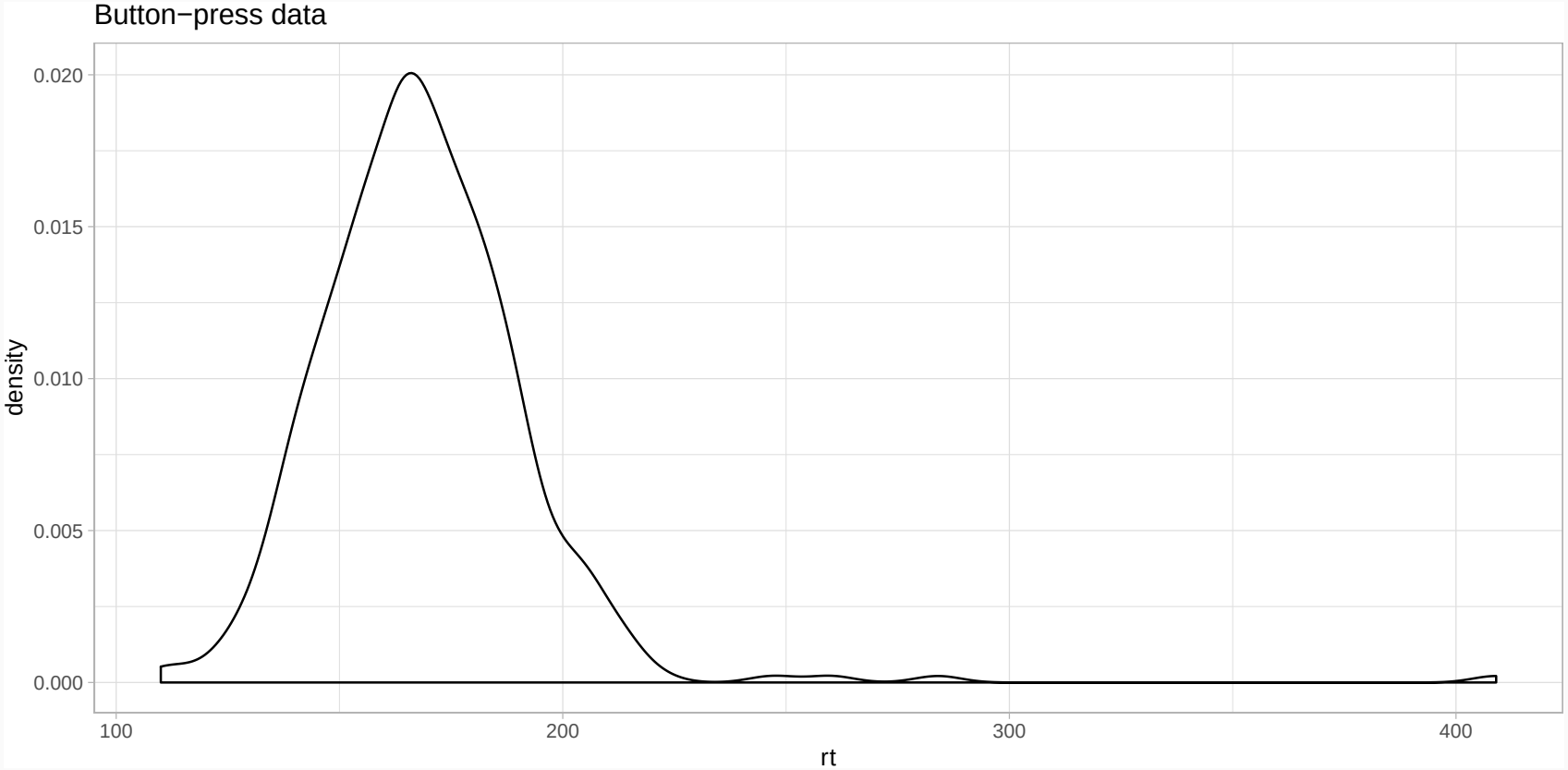


Figure 2: Visualizing the data

Specifying the model in `brms`

```
fit_press <- brm(rt ~ 1,
  data = df_noreading_data,
  family = gaussian(),
  prior = c(
    prior(uniform(0, 60000), class = Intercept),
    prior(uniform(0, 2000), class = sigma)
  ),
  chains = 4,
  iter = 2000,
  warmup = 1000
)
```

Sampling and convergence in a nutshell

1. Chains start in random locations;
2. in each iteration they take one sample each;
3. samples at the beginning do not belong to the posterior distribution;
4. eventually, the chains end up in the vicinity of the posterior distribution;
5. from that point onwards the samples will belong to the posterior.

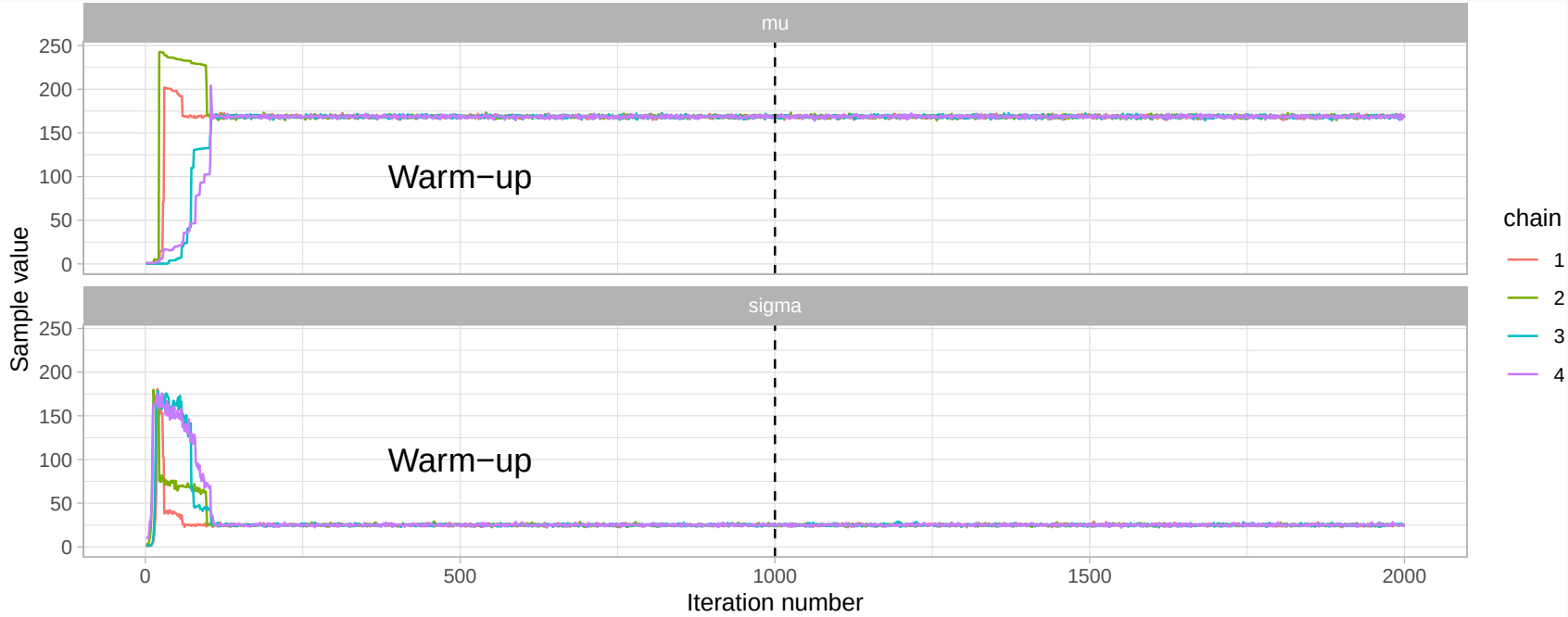


Figure 3: Trace plot of the `brms` model

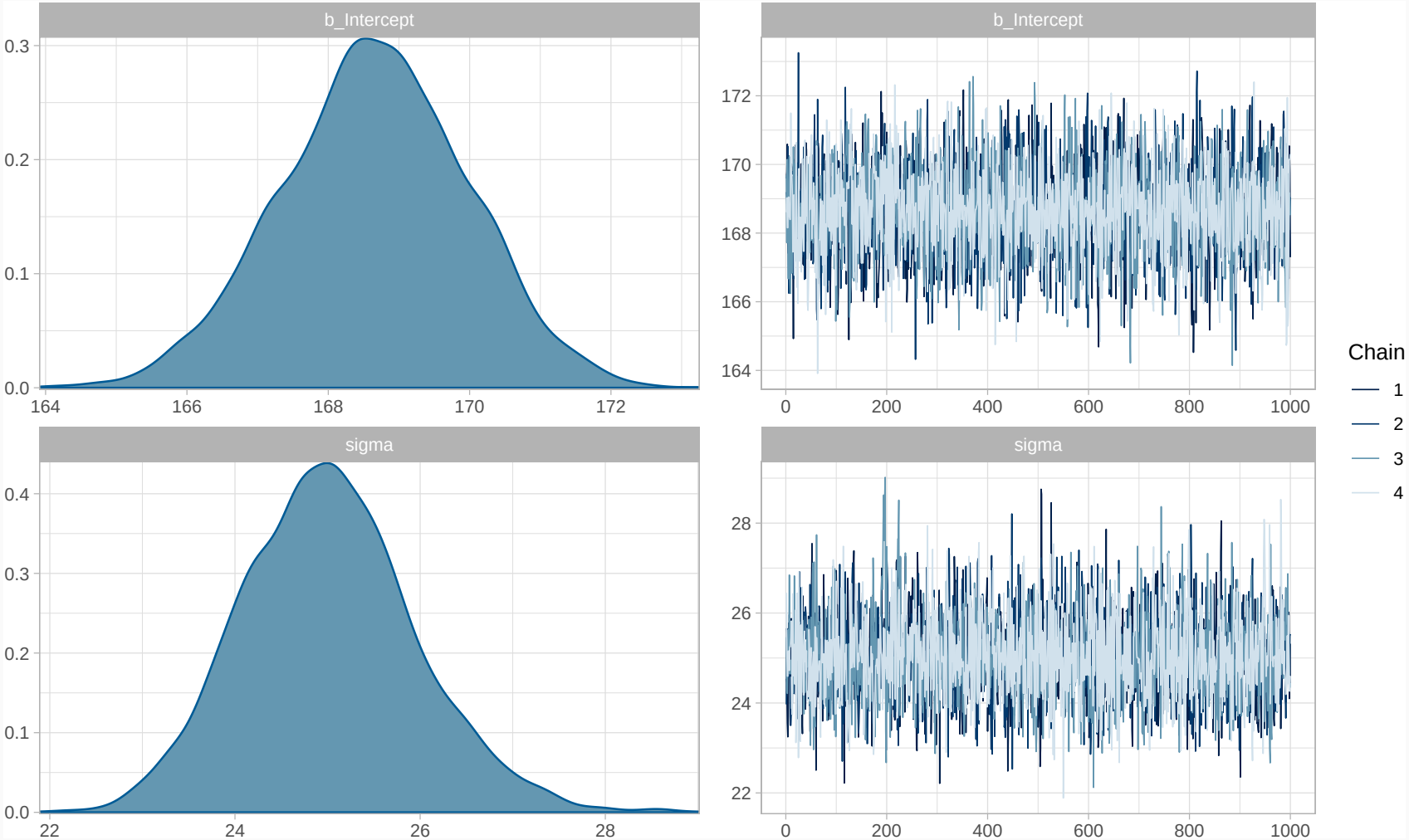
Output of brms

```
posterior_samples(fit_press) %>% str()
```

```
## 'data.frame':    4000 obs. of  3 variables:
##  $ b_Intercept: num  170 170 171 168 167 ...
##  $ sigma      : num  24.5 24.4 23.6 23.2 23.7 ...
##  $ lp__       : num  -1688 -1688 -1690 -1690 -1689 ...
```

Output of brms

```
plot(fit_press)
```



Output of brms

```
fit_press
# posterior_summary(fit_press) is also useful
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: rt ~ 1
##    Data: df_noreading_data (Number of observations: 361)
## 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
## Intercept   168.65      1.32   165.99   171.23 1.00
##           Bulk_ESS Tail_ESS
## Intercept     3474     2661
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat
## sigma    25.01      0.94    23.26    26.99 1.00
##       Bulk_ESS Tail_ESS
## sigma     3528     2398
##
```

Output of brms

Notice that the Estimate is just the mean of the posterior sample, and CI are the 95% quantiles:

```
posterior_samples(fit_press)$b_Intercept %>%
  mean()
```

```
## [1] 169
```

```
posterior_samples(fit_press)$b_Intercept %>%
  quantile(c(0.025, .975))
```

```
## 2.5%  98%
##  166  171
```

Exercises

1. What happens with too few iterations?
2. Using uniform distributions, choose priors that represent better **your** assumptions about reaction times. What happens with the new estimates?

Important questions

1. What information are the priors encoding? Do the priors make sense?
2. Does the likelihood assumed in the model make sense for the data?

Prior predictive distributions

Prior predictive distributions

We want to know the density $p(\cdot)$ of data points $y_1, \ldots, n$, given a vector of priors Θ (e.g., $\Theta = \langle \mu, \sigma \rangle$)

The prior predictive density is:

$$p(y_1, \ldots, y_n) = \int p(y|\Theta) \cdot p(y_2|\Theta) \cdots p(y_n|\Theta) p(\Theta)\, d\Theta \quad (4)$$

We avoid doing the integration by generating samples from the prior distribution. We repeat the following:

1. Take one sample from each of the priors.
2. Plug those samples in the likelihood and generate a dataset $y_{pred_1}, \ldots, y_{pred_n}$.

```r
normal_predictive_distribution <- function(mu_samples, sigma_samples, N_obs) {
  # empty data frame with headers:
  df_pred <- tibble(
    trialn = numeric(0),
    rt_pred = numeric(0),
    iter = numeric(0)
  )
  # i iterates from 1 to the length of mu_samples,
  # which we assume is identical to
  # the length of the sigma_samples:
  for (i in seq_along(mu_samples)) {
    mu <- mu_samples[i]
    sigma <- sigma_samples[i]
    df_pred <- bind_rows(
      df_pred,
      tibble(
        trialn = seq_len(N_obs), # 1, 2,... N_obs
        rt_pred = rnorm(N_obs, mu, sigma),
        iter = i
      )
    )
  }
  df_pred
}
```

This approach works, but it's quite slow:

```
tic()
N_samples <- 1000
N_obs <- nrow(df_noreading_data)
mu_samples <- runif(N_samples, 0, 60000)
sigma_samples <- runif(N_samples, 0, 2000)
normal_predictive_distribution(
  mu_samples = mu_samples,
  sigma_samples = sigma_samples,
  N_obs = N_obs
)
toc()
```

```
## # A tibble: 361,000 x 3
##   trialn rt_pred  iter
##    <dbl>   <dbl> <dbl>
## 1      1  29775.     1
## 2      2  28023.     1
## 3      3  28047.     1
## 4      4  29888.     1
## 5      5  27631.     1
## # ... with 3.61e+05 more rows
## 5.673 sec elapsed
```

A more efficient version:

```r
normal_predictive_distribution_fast <- function(mu_samples,
                                                sigma_samples,
                                                N_obs) {
  # map_dfr works similarly to lapply, it essentially runs
  # a for-loop, and builds a dataframe with the output.
  # We iterate over the values of mu_samples and sigma_samples
  # simultaneously, and in each iteration we bind a new
  # data frame with N_obs observations.
  map2_dfr(mu_samples, sigma_samples, function(mu, sigma) {
    tibble(
      trialn = seq_len(N_obs),
      rt_pred = rnorm(N_obs, mu, sigma)
    )}, .id = "iter") %>%
    # .id is always a string and needs to be converted to a number
    mutate(iter = as.numeric(iter))
}
```

```r
tic()
(prior_pred <- normal_predictive_distribution_fast(
  mu_samples = mu_samples,
  sigma_samples = sigma_samples,
  N_obs))
toc()
```

```
## # A tibble: 361,000 x 3
##    iter trialn rt_pred
##   <dbl>  <int>   <dbl>
## 1     1      1  30582.
## 2     1      2  29972.
## 3     1      3  28733.
## 4     1      4  25289.
## 5     1      5  27985.
## # ... with 3.61e+05 more rows
## 0.473 sec elapsed
```

```r
prior_pred %>%
  filter(iter <= 12) %>%
  ggplot(aes(rt_pred)) +
  geom_histogram() +
  facet_wrap(~iter, ncol = 3)
```

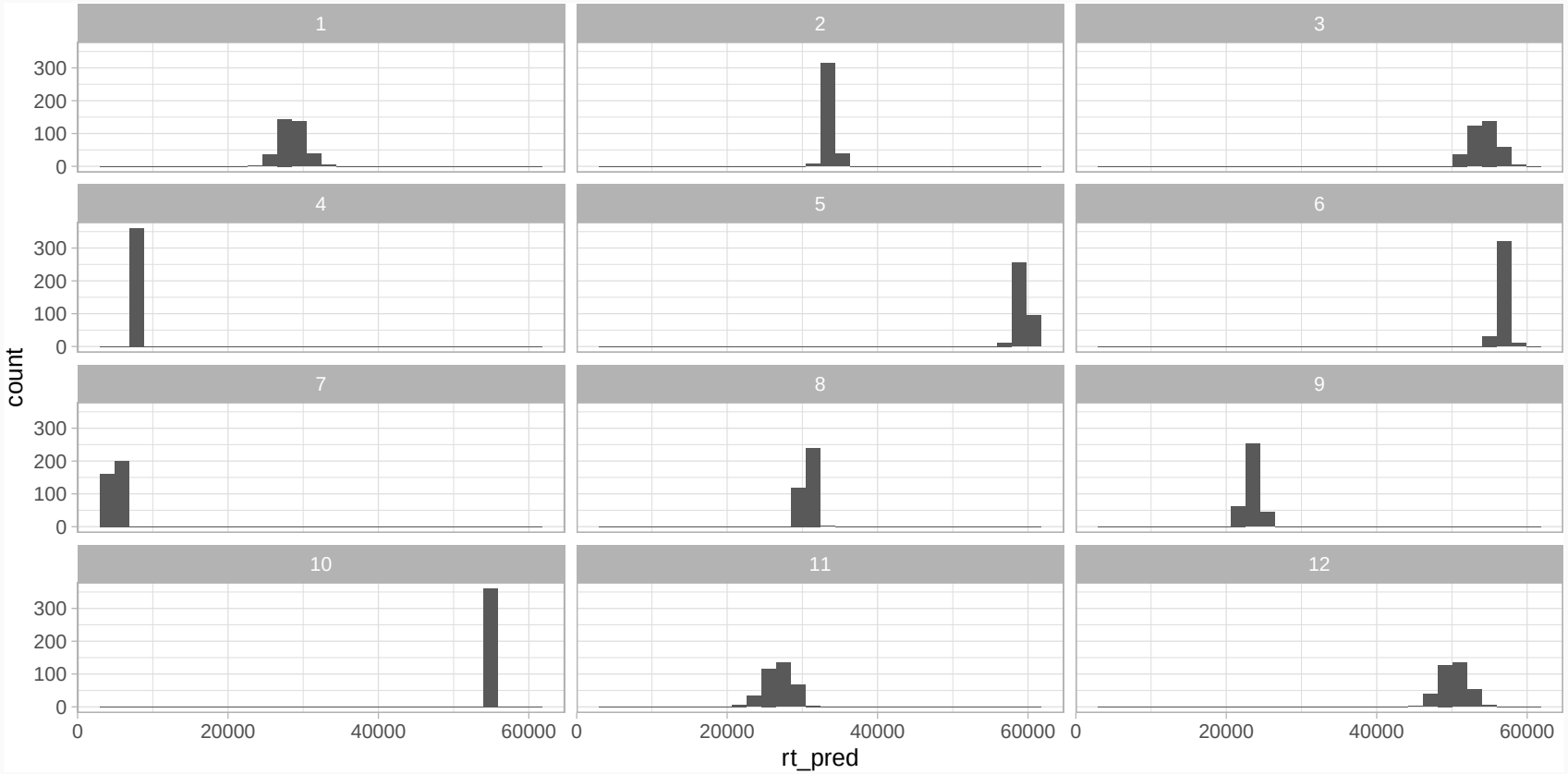


Figure 4: Eighteen samples from the prior predictive distribution.

Distribution of statistics

```
(prior_stat <- prior_pred %>%
  group_by(iter) %>%
  summarize(
    min_rt = min(rt_pred),
    max_rt = max(rt_pred),
    average_rt = mean(rt_pred)
  ) %>%
  # we convert the previous data frame to a long one,
  # where min_rt, max_rt, average_rt are possible values
  # of the columns "stat"
  pivot_longer(
    cols = ends_with("rt"),
    names_to = "stat",
    values_to = "rt"
  ))
```

```
## # A tibble: 3,000 x 3
##    iter stat           rt
##   <dbl> <chr>       <dbl>
## 1     1 min_rt     24127.
## 2     1 max_rt     33359.
```

```
prior_stat %>%
  ggplot(aes(rt)) +
  geom_histogram(binwidth = 500) +
  facet_wrap(~stat, ncol = 1)
```

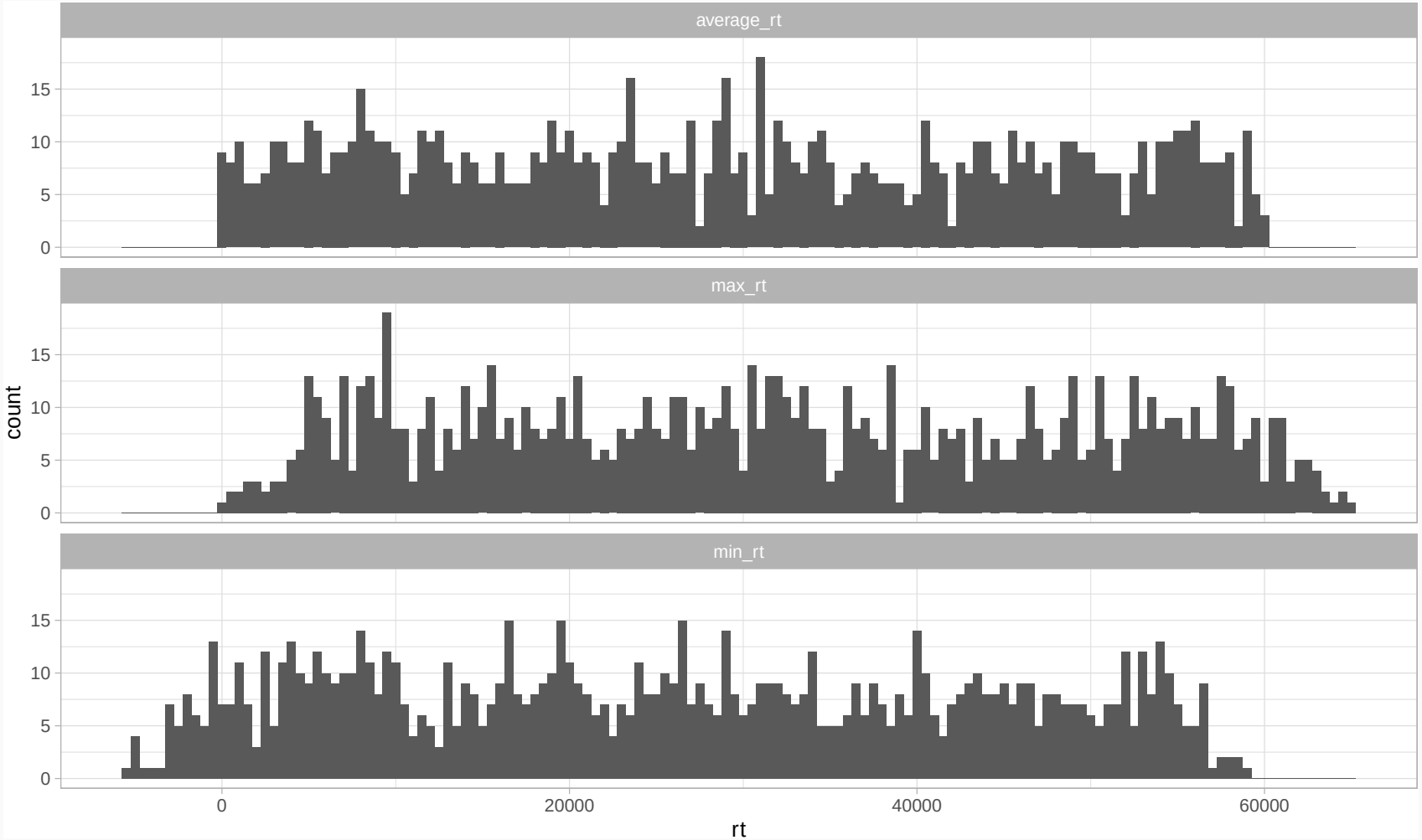


Figure 5: Prior predictive distribution of averages, maximum, and minimum values.

Why are our distributions so bad?

We used much less prior information than what we really had: our priors are clearly not very realistic given what we know about reaction times for such a button pressing task.

What priors should we have chosen?

The influence of priors: sensitivity analysis

Types of priors

1. **Flat uninformative priors**: priors as uninformative as possible.
2. **Regularizing priors**: priors that downweight extreme values (that is, they provide regularization), they are not very informative, and mostly let the likelihood dominate in determining the posteriors.
3. **Principled priors**: priors that encode all (or most of) the theory-neutral information that we do have.
4. **Informative priors**: There are cases where we have a lot of prior knowledge, and not much data.

Revisiting the button-pressing example with different priors

What would happen if we use even wider priors for the model?

$$\begin{aligned} \mu &\sim Uniform(-10^{10}, 10^{10}) \\ \sigma &\sim Uniform(0, 10^{10}) \end{aligned} \tag{5}$$

In brms:

```
fit_press_unif <- brm(rt ~ 1,
  data = df_noreading_data,
  family = gaussian(),
  prior = c(
      prior(uniform(-10^10, 10^10), class = Intercept),
    prior(uniform(0, 10^10), class = sigma))
)
```

The output of the model is virtually identical!

```
fit_press_unif
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: rt ~ 1
##    Data: df_noreading_data (Number of observations: 361)
## 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
## Intercept   168.62      1.33   165.94   171.25 1.00
##           Bulk_ESS Tail_ESS
## Intercept     3709     2105
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat
## sigma    24.98      0.91    23.35    26.84 1.00
##       Bulk_ESS Tail_ESS
## sigma     3592     2830
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
```

What happens if we use very informative priors and they are off?

$$\mu \sim Normal(400, 10)$$
$$\sigma \sim Normal_{+}(100, 10) \tag{6}$$

```
fit_press_inf <- brm(rt ~ 1,
  data = df_noreading_data,
  family = gaussian(),
  prior = c(
    prior(normal(400, 10), class = Intercept),
    # brms knows that SD needs to be bounded by zero:
    prior(normal(100, 10), class = sigma)
  )
)
```

Even in this case, the new estimates are just a couple of milliseconds away from our previous estimates:

```
fit_press_inf
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: rt ~ 1
##    Data: df_noreading_data (Number of observations: 361)
## 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
## Intercept   172.93      1.44   170.11   175.83 1.00
##           Bulk_ESS Tail_ESS
## Intercept     2426     2066
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat
## sigma    26.09      1.04    24.11    28.20 1.00
##       Bulk_ESS Tail_ESS
## sigma     2604     2582
##
```

This doesn't mean that priors never matter:

- When there is enough data for *a certain parameter*, the likelihood will dominate
- If we are not sure about the extent to which the posterior is influenced by our priors, we can do a *sensitivity analysis* (for a published example in psycholinguistics, see Vasishth et al. 2013).
- We can use prior predictive distributions to see if we are on the right order of magnitude for our priors

Exercises

3. Can you come up with very informative priors that bias the posterior in a noticeable way (using normally distributed priors)? Generate a prior predictive distribution based on this prior.

Posterior predictive distributions

Once we have the posterior distribution $p(\Theta \mid y)$, we can derive the predictions based on this distribution:

$$p(D_{pred} \mid y) = \int_{\Theta} p(D_{pred} \mid \Theta) p(\Theta \mid y)\, d\Theta \tag{7}$$

We can also here avoid the integration, and we can even use the same function that we created before:

```
N_obs <- nrow(df_noreading_data)
mu_samples <- posterior_samples(fit_press)$b_Intercept
sigma_samples <- posterior_samples(fit_press)$sigma
(normal_predictive_distribution_fast(
  mu_samples = mu_samples,
  sigma_samples = sigma_samples,
  N_obs
))
```

```
## # A tibble: 1,444,000 x 3
##    iter trialn rt_pred
##   <dbl>  <int>   <dbl>
## 1     1      1    152.
## 2     1      2    182.
## 3     1      3    138.
## 4     1      4    198.
## 5     1      5    178.
## # ... with 1.444e+06 more rows
```

Descriptive adequacy/posterior predictive checks

Could the current data have been generated by our model?

```
pp_check(fit_press, nsamples = 11, type = "hist")
```

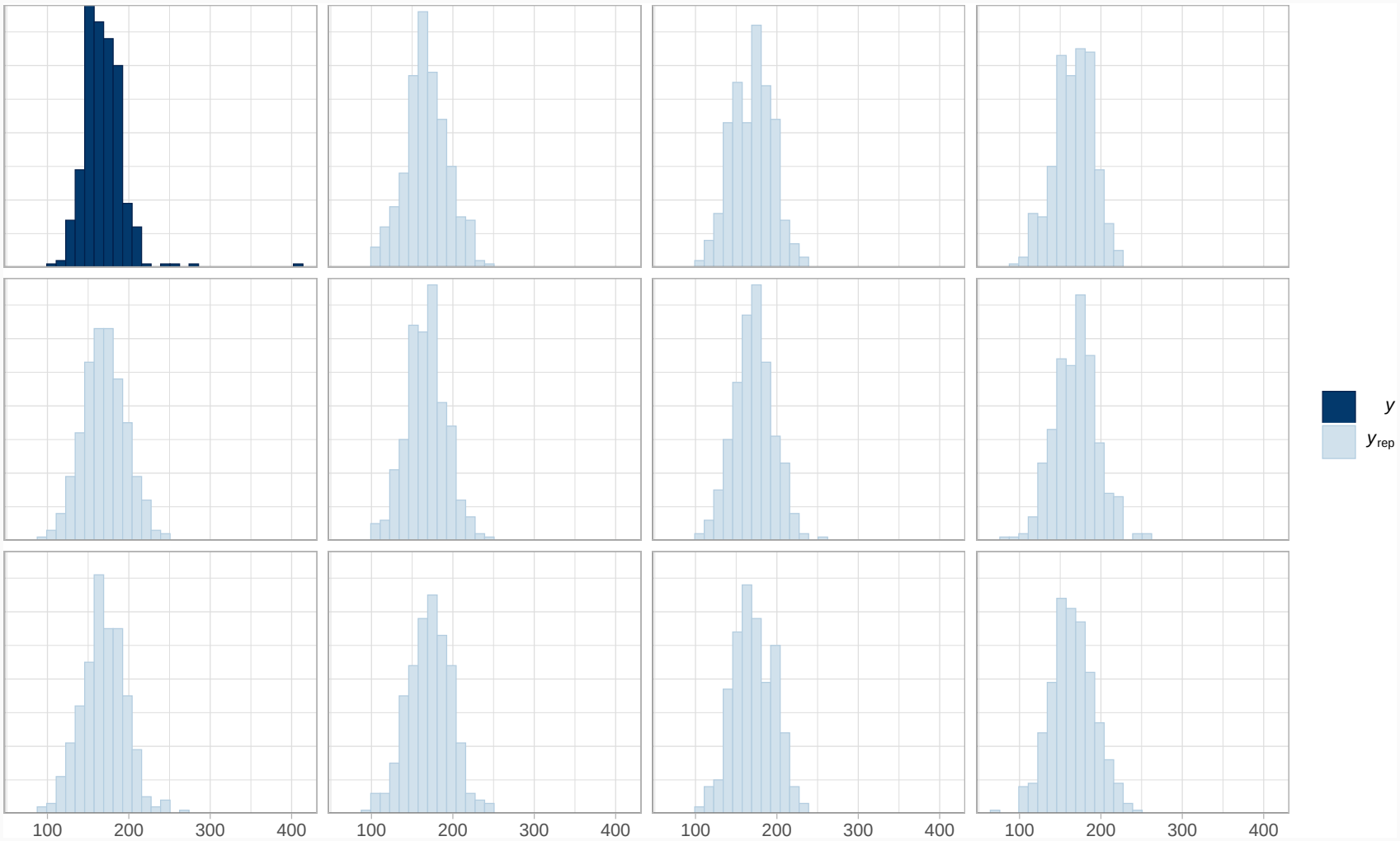


Figure 6: Eleven samples from the posterior predictive distribution of the model `fit_press`.

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

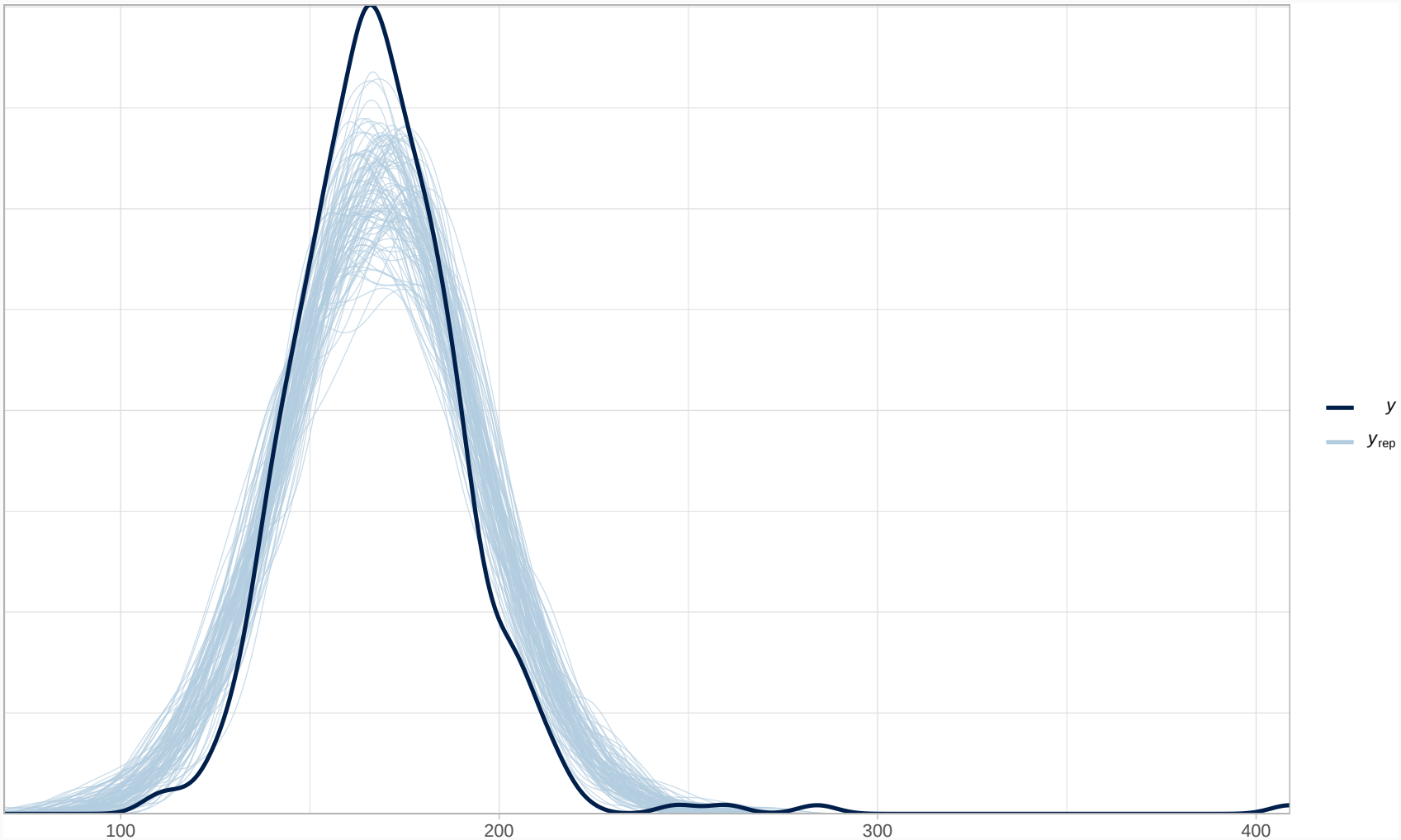


Figure 7: Posterior predictive check that shows the fit of the model `fit_press` in comparison to datasets from the posterior predictive distribution.

Comparing different likelihoods: The log-normal likelihood

If y is log-normally distributed, this means that $\log(y)$ is normally distributed.[2]

$$\begin{aligned} \log(y) &\sim Normal(\mu, \sigma) \\ y &\sim \exp(Normal(\mu, \sigma)) \\ y &\sim LogNormal(\mu, \sigma) \end{aligned} \tag{8}$$

The log-normal distribution is again defined using μ and σ, but these correspond to the mean and standard deviation of the normally distributed logarithm of the data y: $\log(y)$.

[2] In fact, $\log_e(y)$ or $\ln(y)$, but we'll write it as just $log()$

Re-fitting a single participant pressing a button repeatedly with a log-normal likelihood

New likelihood:

$$rt_n \sim LogNormal(\mu, \sigma) \tag{9}$$

New scale for the priors:

$$\begin{aligned} \mu &\sim Uniform(0, 8) \\ \sigma &\sim Uniform(0, 1) \end{aligned} \tag{10}$$

Because the parameters are in a different scale than the dependent variable, their interpretation changes:

- *The location,* μ: In our previous linear model, μ represented the grand mean (or the grand median, or grand mode, since in a normal distribution the three coincide). But now, the grand mean is $\exp(\mu + \sigma^2/2)$ and the grand median is $\exp(\mu)$.
- *The scale,* σ: This is the standard deviation of the normal distribution of $\log(y)$. The standard deviation of a log-normal distribution with *location* μ and *shape* σ will be $\exp(\mu + \sigma^2/2) \times \sqrt{(\exp(\sigma^2) - 1)}$.

Prior predictive distributions

```
N_samples <- 1000
N_obs <- nrow(df_noreading_data)
mu_samples <- runif(N_samples, 0, 8)
sigma_samples <- runif(N_samples, 0, 1)
prior_pred_ln <- exp(normal_predictive_distribution_fast(
  mu_samples = mu_samples,
  sigma_samples = sigma_samples,
  N_obs
))
```

Distribution of statistics

```
(prior_pred_stat_ln <-
  prior_pred_ln %>%
  group_by(iter) %>%
  summarize(
    min_rt = min(rt_pred),
    max_rt = max(rt_pred),
    average_rt = mean(rt_pred),
    median_rt = median(rt_pred)
  ) %>%
  pivot_longer(cols = ends_with("rt"), names_to = "stat", values_to = "rt"))
```

```
## # A tibble: 2,840 x 3
##    iter stat          rt
##   <dbl> <chr>      <dbl>
## 1  2.72 min_rt      4.51
## 2  2.72 max_rt      5.69
## 3  2.72 average_rt  5.06
## 4  2.72 median_rt   5.05
## 5  7.39 min_rt      5.85
## # ... with 2,835 more rows
```

```r
prior_pred_stat_ln %>%
  ggplot(aes(rt)) +
  scale_x_continuous("Reaction times in ms",
    trans = "log", breaks = c(0.001, 1, 100, 1000, 10000, 100000)
  ) +
  geom_histogram() +
  facet_wrap(~stat, ncol = 1)
```

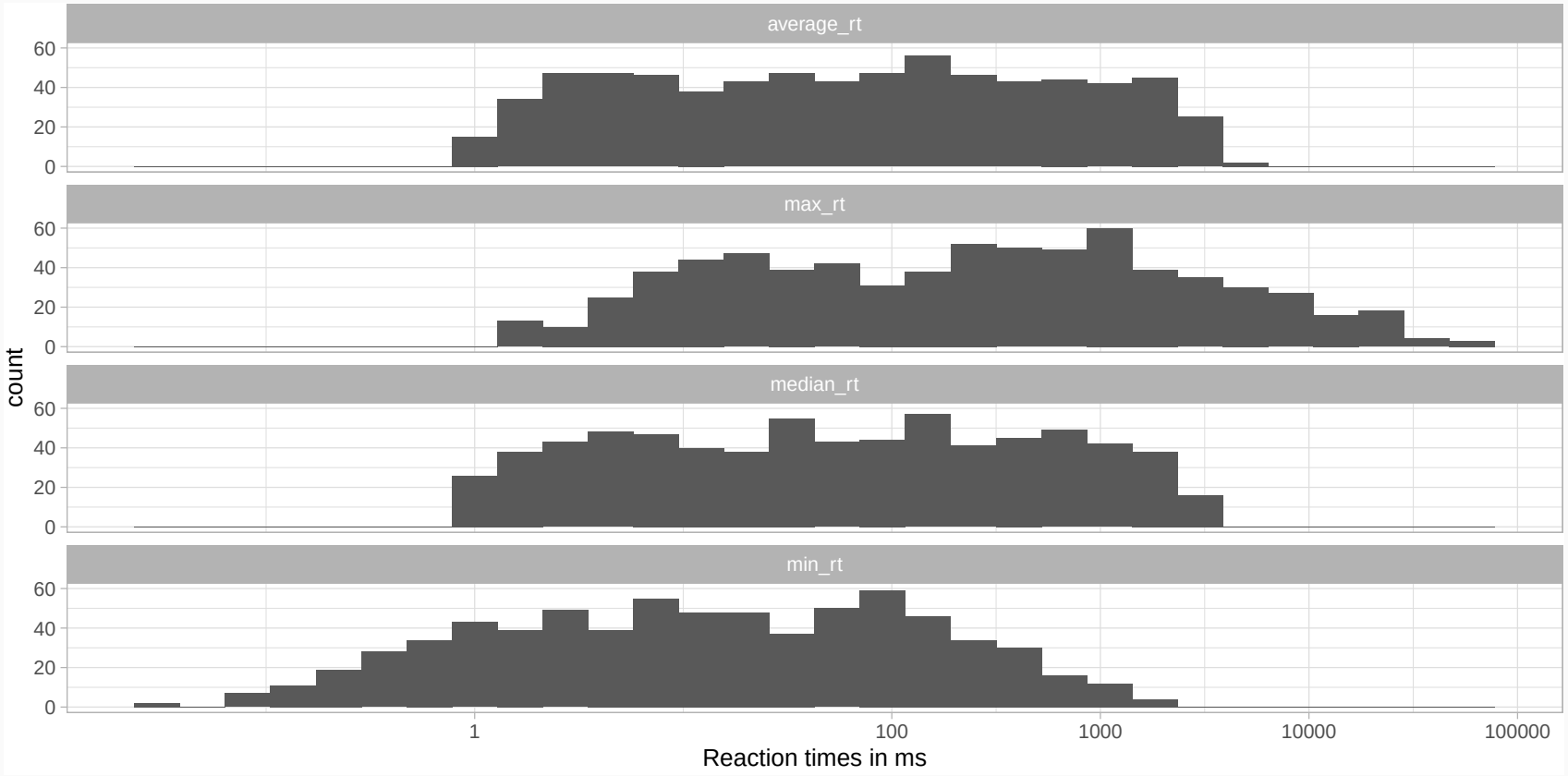


Figure 8: Prior predictive distribution of averages, maximum, and minimum value of the log-normal model; the x-axis is log-transformed.

Better regularizing priors for the log-normal model

$$rt_n \sim LogNormal(\mu, \sigma) \tag{11}$$

$$\begin{aligned} \mu &\sim Normal(6, 1.5) \\ \sigma &\sim Normal_+(0, 1) \end{aligned} \tag{12}$$

Median effect for our new priors:

```
c(
  lower = exp(6 - 2 * 1.5),
  higher = exp(6 + 2 * 1.5)
)
```

```
##  lower higher
##     20   8103
```

Prior predictive distributions

```
N_samples <- 1000
N_obs <- nrow(df_noreading_data)
mu_samples <- rnorm(N_samples, 6, 1.5)
sigma_samples <- rtnorm(N_samples, 0, 1, a = 0)
(prior_pred_ln_better <- exp(normal_predictive_distribution_fast(
  mu_samples = mu_samples,
  sigma_samples = sigma_samples,
  N_obs
)))
```

```
## # A tibble: 361,000 x 3
##    iter trialn rt_pred
##   <dbl>  <dbl>   <dbl>
## 1  2.72   2.72   4354.
## 2  2.72   7.39   8813.
## 3  2.72  20.1    6651.
## 4  2.72  54.6    8723.
## 5  2.72 148.     5968.
## # ... with 3.61e+05 more rows
```

```
(prior_pred_stat_better_ln <- prior_pred_ln_better %>%
  group_by(iter) %>%
  summarize(
    min_rt = min(rt_pred),
    max_rt = max(rt_pred),
    average_rt = mean(rt_pred),
    median_rt = median(rt_pred)
  ) %>%
  pivot_longer(
    cols = ends_with("rt"),
    names_to = "stat", values_to = "rt"
  ))
```

```
## # A tibble: 2,840 x 3
##    iter stat           rt
##   <dbl> <chr>       <dbl>
## 1  2.72 min_rt      2357.
## 2  2.72 max_rt     11945.
## 3  2.72 average_rt  5822.
## 4  2.72 median_rt   5557.
## 5  7.39 min_rt      1718.
## # ... with 2,835 more rows
```

```r
prior_pred_stat_better_ln %>% ggplot(aes(rt)) +
    scale_x_continuous(trans = "log",
                       breaks = c(0.001, 1, 100, 1000, 10000, 100000)) +
    geom_histogram() +
    facet_wrap(~stat, ncol = 1) +
    coord_cartesian(xlim = c(0.001, 300000))
```

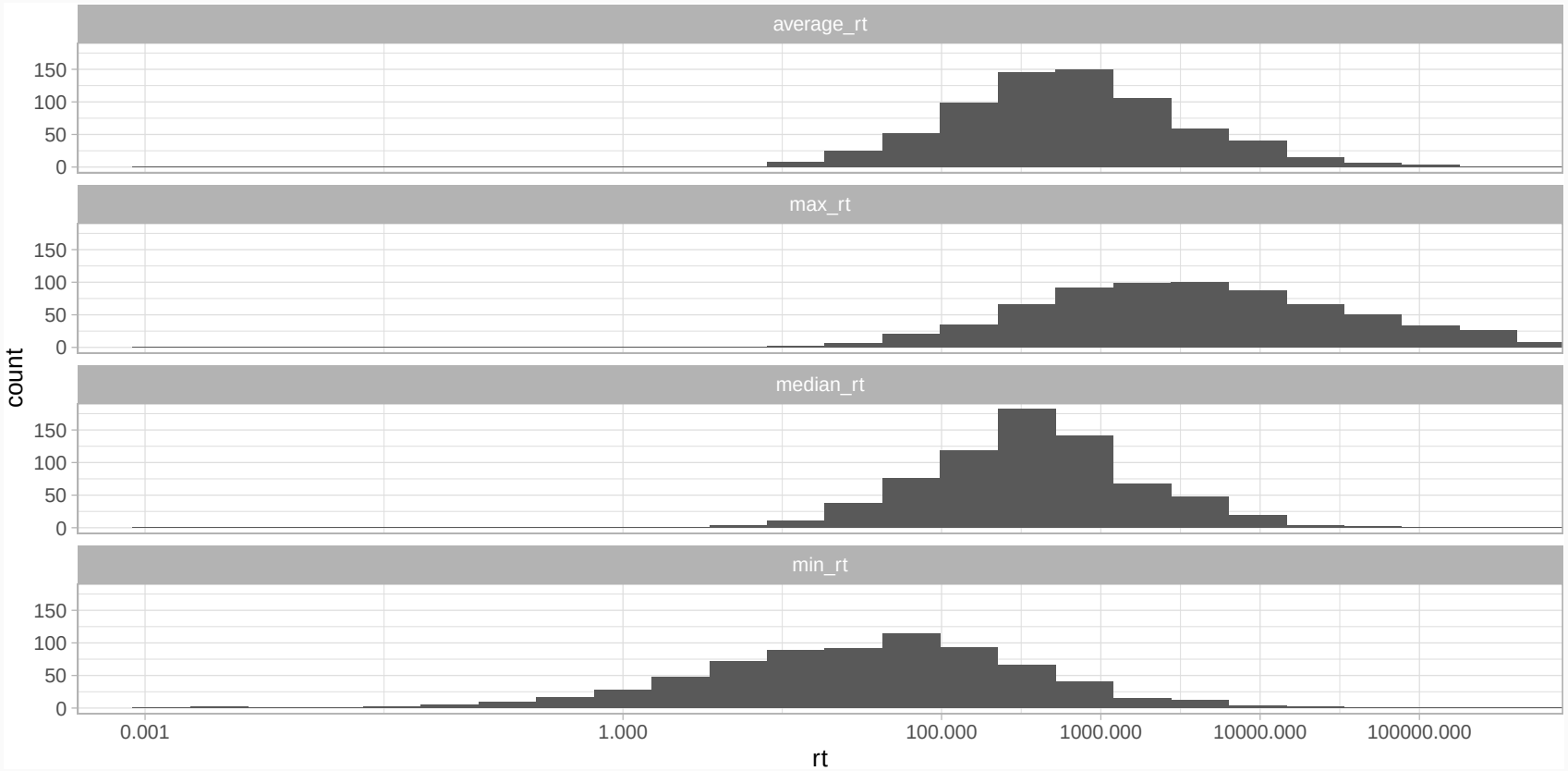


Figure 9: Prior predictive distribution of averages, maximum, and minimum value of the log-normal model with better priors.

brms model with reasonable priors:[3]

```
fit_press_ln <- brm(rt ~ 1,
  data = df_noreading_data,
  family = lognormal(),
  prior = c(
    prior(normal(6, 1.5), class = Intercept),
    prior(normal(0, 1), class = sigma)
  )
)
```

[3] Notice that we need to specify that the family is `lognormal()`

```
fit_press_ln
```

```
##  Family: lognormal
##   Links: mu = identity; sigma = identity
## Formula: rt ~ 1
##    Data: df_noreading_data (Number of observations: 361)
## 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
## Intercept     5.12      0.01     5.10     5.13 1.00
##           Bulk_ESS Tail_ESS
## Intercept     3395     2602
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat
## sigma     0.13      0.01     0.13     0.15 1.00
##       Bulk_ESS Tail_ESS
## sigma     2534     1994
##
## 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).
```

How long does it take to press the space bar in milliseconds?

```
estimate_ms <- exp(posterior_samples(fit_press_ln)$b_Intercept)
c(mean = mean(estimate_ms), quantile(estimate_ms, probs = c(.025, .975)))
```

```
## mean 2.5%  98%
##  167  165  169
```

Posterior predictive checks

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

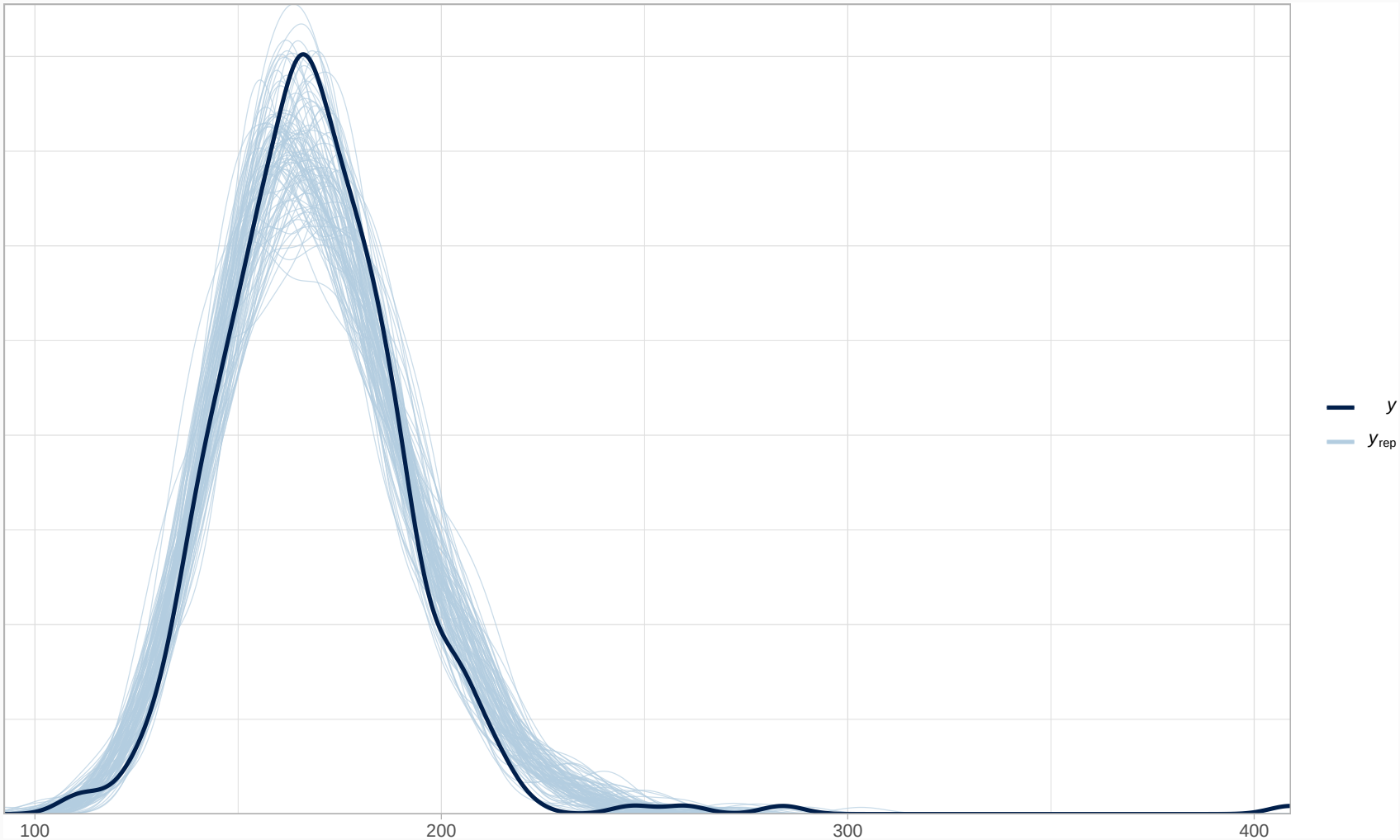


Figure 10: Posterior predictive distribution of `fit_noreading_ln`

Are the posterior predicted data now more similar to the real data, compared to the case where we had a Normal likelihood?

We suspect that the normal distribution would generate reaction times that are too fast (since it's symmetrical) and that the log-normal distribution may capture the long tail better than the normal model.

```
pp_check(fit_press, type = "stat", stat = "min") +
  ggtitle("Normal model")
pp_check(fit_press_ln, type = "stat", stat = "min") +
  ggtitle("Log-normal model")
```

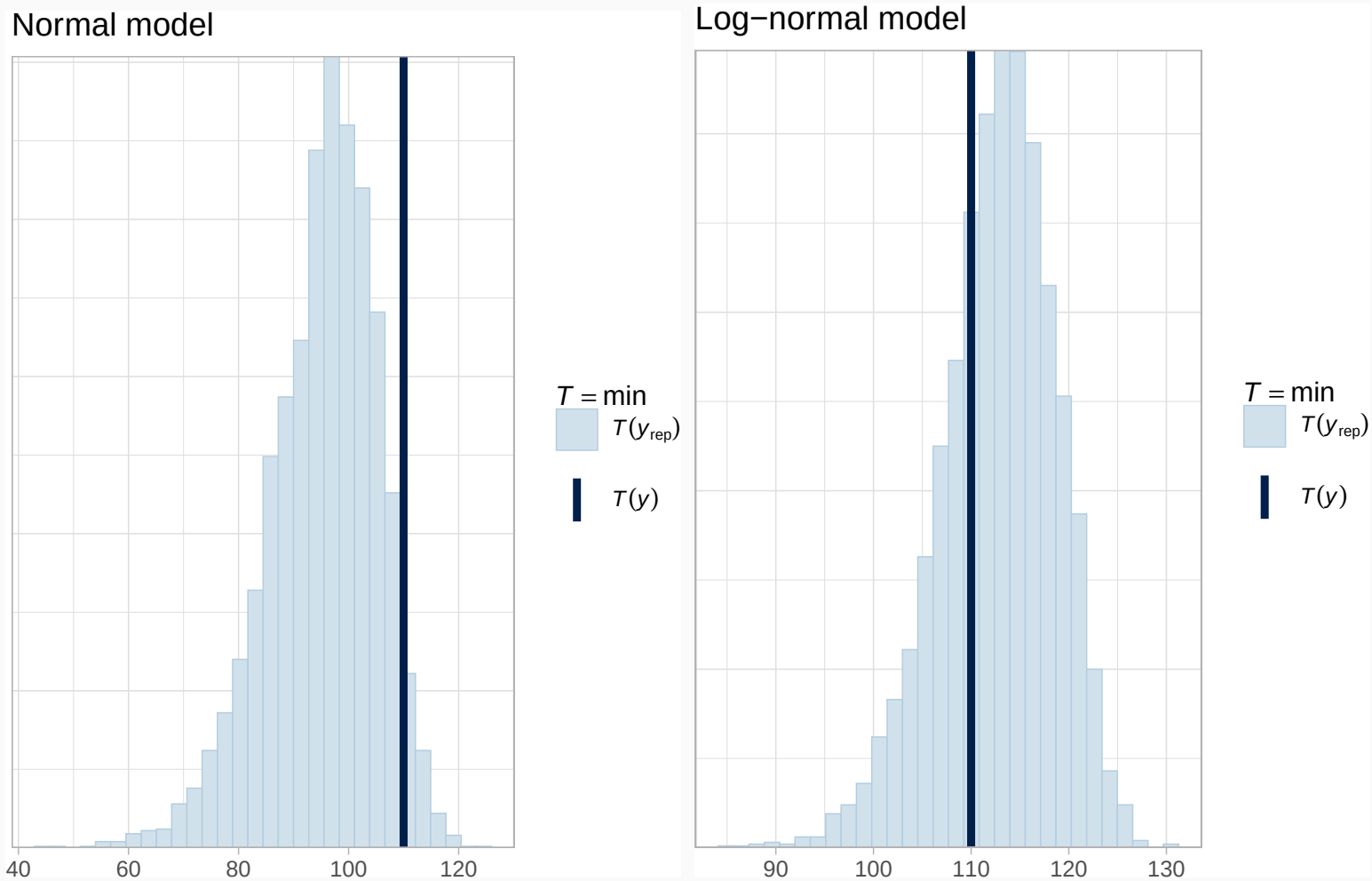


Figure 11: Distribution of minimum values in a posterior predictive check. The minimum in the data is 110 ms.

```
pp_check(fit_press, type = "stat", stat = "max") +
  ggtitle("Normal model")
pp_check(fit_press_ln, type = "stat", stat = "max") +
  ggtitle("Log-normal model")
```

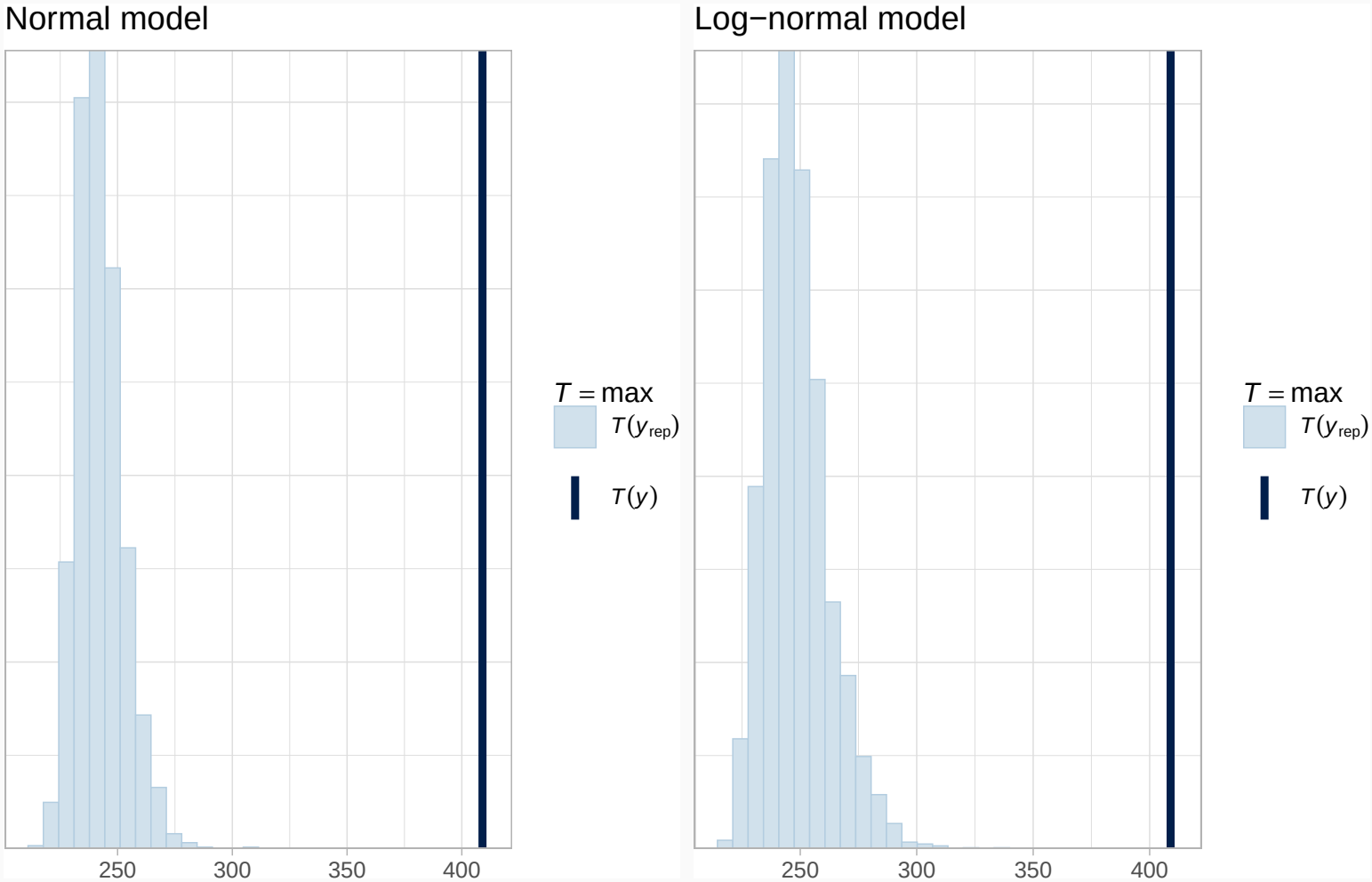


Figure 12: Distribution of maximum values in a posterior predictive check. The maximum in the data is 409 ms.

Exercises

4. Change the prior of σ to .5 and generate prior predictive distributions.
5. Try to find sensible priors with prior predictive checks.
6. What is the mean (rather than median) time that takes to press the space bar, what is the standard deviation of the reaction times in milliseconds?

What did we do?

- fitted and interpreted a normal model
- looked at the effect of priors:
  - prior predictive distributions
  - sensitivity analysis
- looked at the fit of the posterior:
  - posterior predictive distribution (descriptive adequacy)
- fitted and interpreted a log-normal model
- compared a normal model with a log-normal one

References

Bürkner, Paul-Christian. 2019. *Brms: Bayesian Regression Models Using 'Stan'*. https://CRAN.R-project.org/package=brms.

Carpenter, Bob, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. "Stan: A Probabilistic Programming Language." *Journal of Statistical Software* 76 (1). Columbia Univ., New York, NY (United States); Harvard Univ., Cambridge, MA (United States).

Goodrich, Ben, Jonah Gabry, Imad Ali, and Sam Brilleman. 2018. "Rstanarm: Bayesian Applied Regression Modeling via Stan." http://mc-stan.org/.