HW 2 Solutions

Shravan Vasishth

```
## loads all the packages needed:
source("index.R")
```

Exercise 1

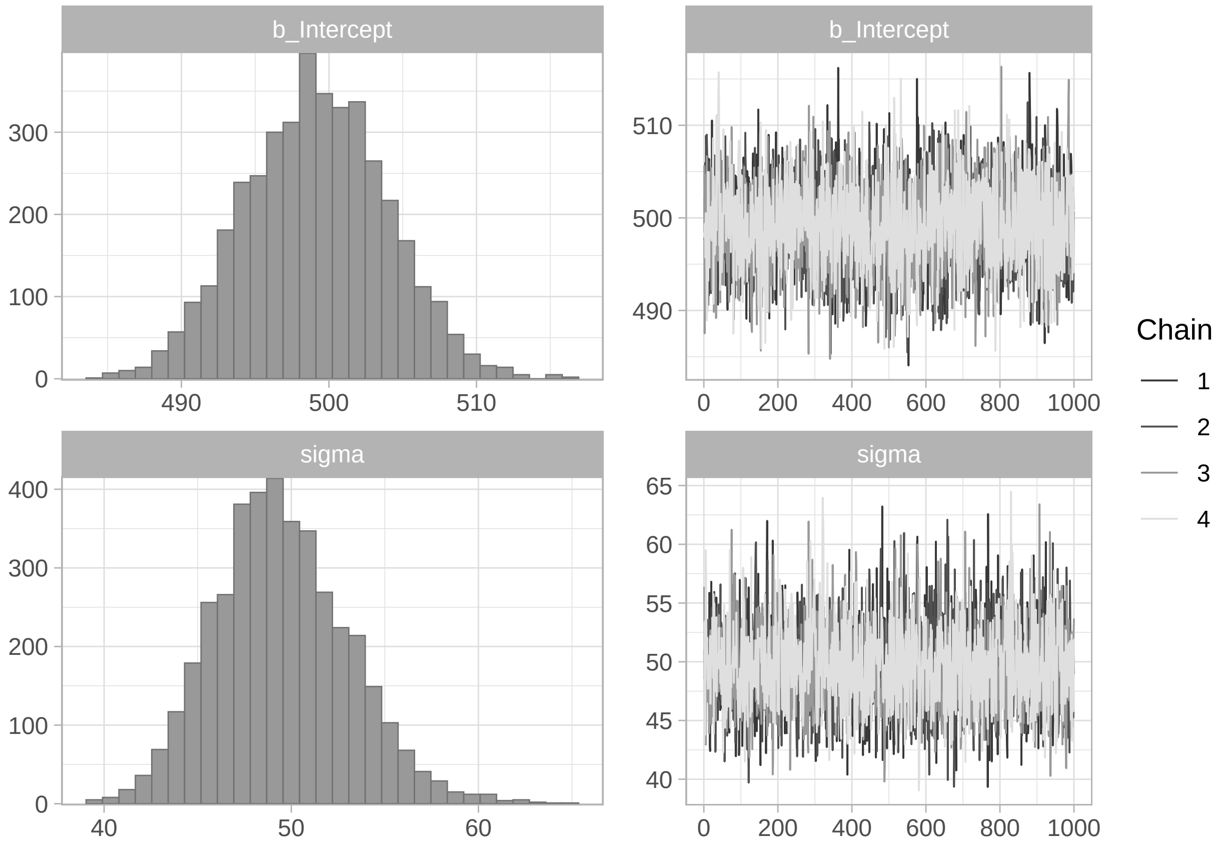
Generate some simulated data with known parameter values:

```
y<-rnorm(100,mean=500,sd=50)
sim_dat<-data.frame(y=y)
```

```
sim_model <- brm(y ~ 1,
  data = sim_dat,
  family = gaussian(),
  prior = c(
    prior(uniform(0, 60000), class = Intercept, lb = 0, ub = 60000),
    prior(uniform(0, 2000), class = sigma, lb = 0, ub = 2000)
  ),
  chains = 4,
  iter = 2000,
  warmup = 1000
)
```

Look at the posterior distributions:

```
plot(sim_model)
```



Confirm through visual inspect that the true values ($\mu = 500, \sigma = 50$) do fall within the respective posterior distributions shown in the plot.

Exercise 2

(a) Fit the model fit_press with just a few iterations, say 50 iterations. What happens?

```
data("df_spacebar")
```

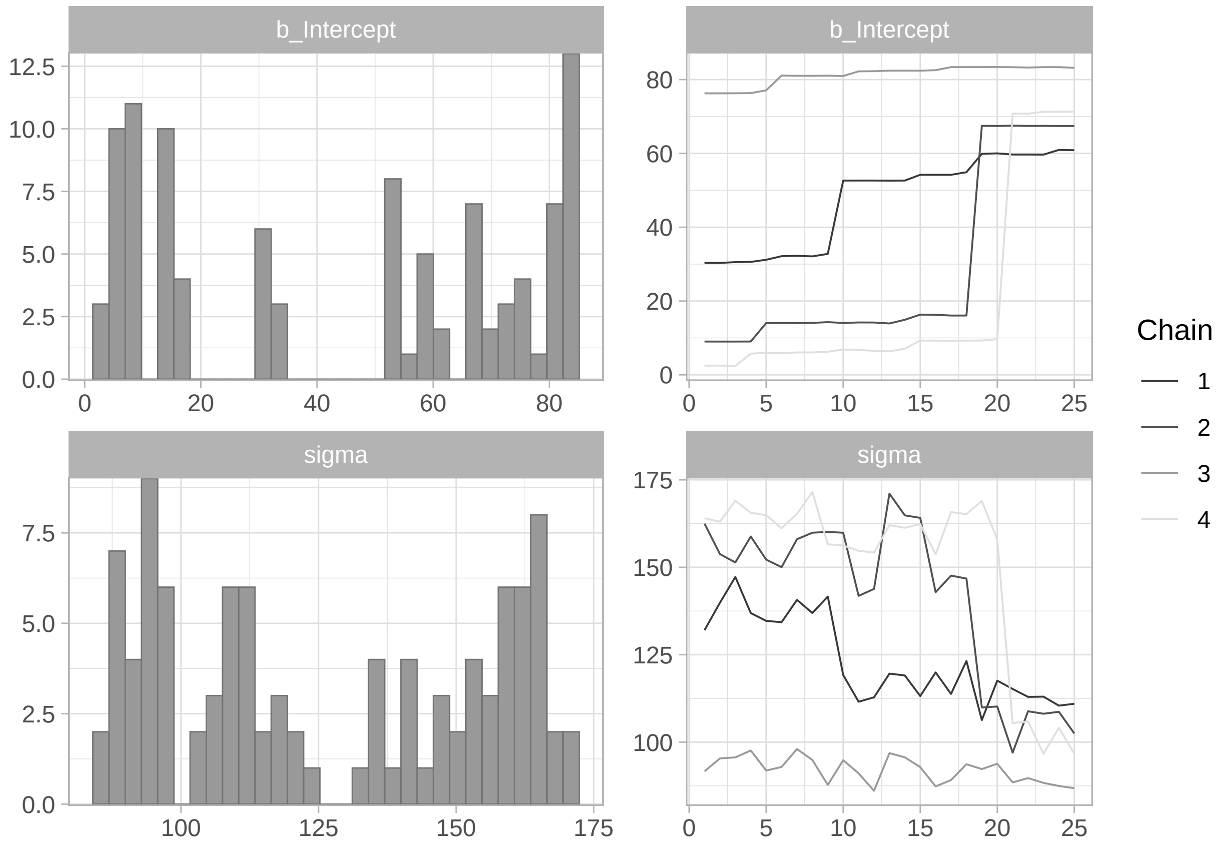
We change the model so that `iter = 50` (the warmup will by default be 25):

```
fit_press_bad <- brm(t ~ 1,
  data = df_spacebar,
  family = gaussian(),
  prior = c(
    prior(uniform(0, 60000), class = Intercept),
    prior(uniform(0, 2000), class = sigma)
  ),
  chains = 4,
  iter = 50
)
```

```
## Warning: It appears as if you have specified a lower bounded prior on a parameter that has no natural
## If this is really what you want, please specify argument 'lb' of 'set_prior' appropriately.
## Warning occurred for prior
## Intercept ~ uniform(0, 60000)
## Warning: It appears as if you have specified an upper bounded prior on a parameter that has no natura
## If this is really what you want, please specify argument 'ub' of 'set_prior' appropriately.
## Warning occurred for prior
## Intercept ~ uniform(0, 60000)
## <lower=0> sigma ~ uniform(0, 2000)

## Warning: There were 4 chains where the estimated Bayesian Fraction of Missing Information was low. Se
## https://mc-stan.org/misc/warnings.html#bfmi-low

## Warning: Examine the pairs() plot to diagnose sampling problems

## Warning: The largest R-hat is NA, indicating chains have not mixed.
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#r-hat

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quant
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

This is definitely a bad idea. As it is clear from all the warnings, the number of iterations is not enough to achieve convergence. Let's also take a look at the traceplots:

```
plot(fit_press_bad)
```



We see that the chains did not mix in the few iterations that were run. Notice that the warning messages convey important information, they suggest the following: `Running the chains for more iterations may help`.

(b) Using normal distributions, choose priors that better represent your assumptions about response times.

Here is a plausible set of informative priors:

$$\begin{aligned} \mu &\sim Normal(200, 100) \\ \sigma &\sim Normal_{+}(50, 50) \end{aligned} \tag{1}$$

```r
fit_press_personal <- brm(t ~ 1,
  data = df_spacebar,
  family = gaussian(),
  prior = c(
    prior(normal(200, 100), class = Intercept),
    prior(normal(50, 50), class = sigma)
  )
)
```

Exercise 3

a. Can you come up with very informative priors that bias the posterior in a noticeable way (use normal distributions for priors, not uniform priors)?

Coming up with very informative priors that bias the posterior in a noticeable way is actually not that easy; one would need to include a lot of unreasonable certainty in the priors to achieve this. For example, see the model shown below:

```r
fit_press_too_inf <- brm(t ~ 1,
  data = df_spacebar,
  family = gaussian(),
  prior = c(
    prior(normal(400, 1), class = Intercept,lb=0),
    prior(normal(100, 10), class = sigma, lb=0)
  )
)
```

```r
fit_press_too_inf
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: t ~ 1
##    Data: df_spacebar (Number of observations: 361)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Regression Coefficients:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept   397.68      1.01   395.73   399.70 1.00     3853     2696
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma   190.07      4.89   180.78   199.88 1.00     1745     2111
##
## Draws were sampled 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).
```

b. Generate and plot prior predictive distributions based on this prior.

Here is the prior predictive distribution based on the above priors:

Option 1: Generating the prior predictive distribution with `brms`:

```r
fit_press_too_inf_prior <- brm(t ~ 1,
  data = df_spacebar,
  family = gaussian(),
  prior = c(
    prior(normal(400, 1), class = Intercept),
    prior(normal(100, 10), class = sigma)
  ),
  sample_prior = "only"
)
```

Plot the prior predictive distributions using `pp_check`:

```r
pp_check(fit_press_too_inf_prior, ndraws = 14, type = "hist")
```

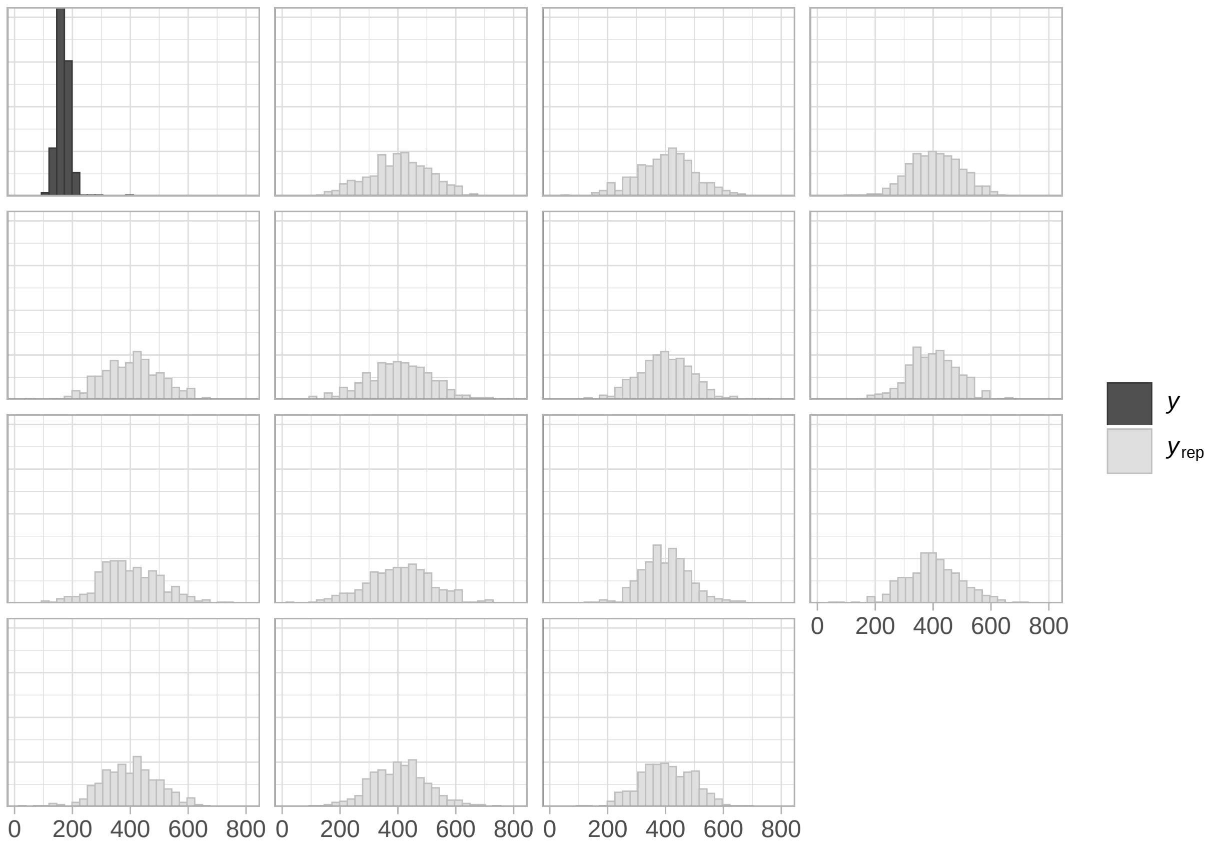


Figure 1: Eighteen samples from the prior predictive distribution of the model with too informative priors shown as `yrep`; `y` is the observed data

Option 2: Generating the prior predictive distribution by "hand":

This option is more involved but gives us more flexibility. We use `map2_dfr` from the package `purrr`.

```r
normal_predictive_distribution_fast <- function(mu_samples,
                                                sigma_samples,
                                                N_obs) {
```

```
  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))
}


N_samples <- 1000
N_obs <- nrow(df_spacebar)
mu_samples_too_inf <- rnorm(N_samples, 400, 1)
sigma_samples_too_inf <- rtnorm(N_samples, 100, 10, a = 0)

prior_pred_too_inf <- normal_predictive_distribution_fast(
  mu_samples = mu_samples_too_inf,
  sigma_samples = sigma_samples_too_inf,
  N_obs
)
```

Plot the distributions to evaluate them:

```
prior_pred_too_inf %>%
  filter(iter <= 18) %>%
  ggplot(aes(rt_pred)) +
  geom_histogram() +
  facet_wrap(~iter, ncol = 3)
```

One can also look at the distribution of statistics:

```
prior_pred_too_inf %>%
  group_by(iter) %>%
  summarize(
    min_rt = min(rt_pred),
    max_rt = max(rt_pred),
    average_rt = mean(rt_pred)
  ) %>%
  pivot_longer(
    cols = ends_with("rt"),
    names_to = "stat",
    values_to = "rt"
  ) %>%
  ggplot(aes(rt)) +
  geom_histogram(binwidth = 20) +
  facet_wrap(~stat, ncol = 1)
```

The prior predictive distributions show that the priors are clearly too strong and informative restricting the possible values of μ too much.

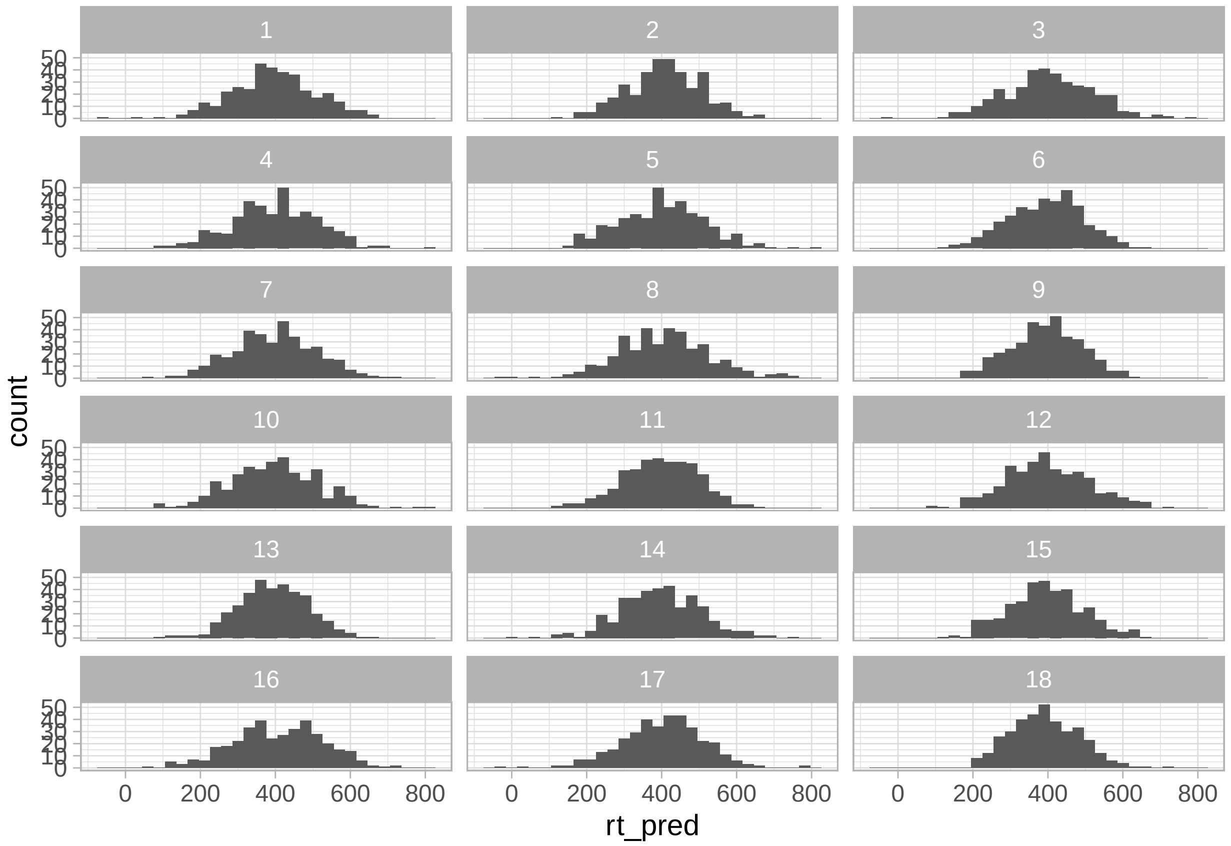


Figure 2: Eighteen samples from the prior predictive distribution of the model with too informative priors.

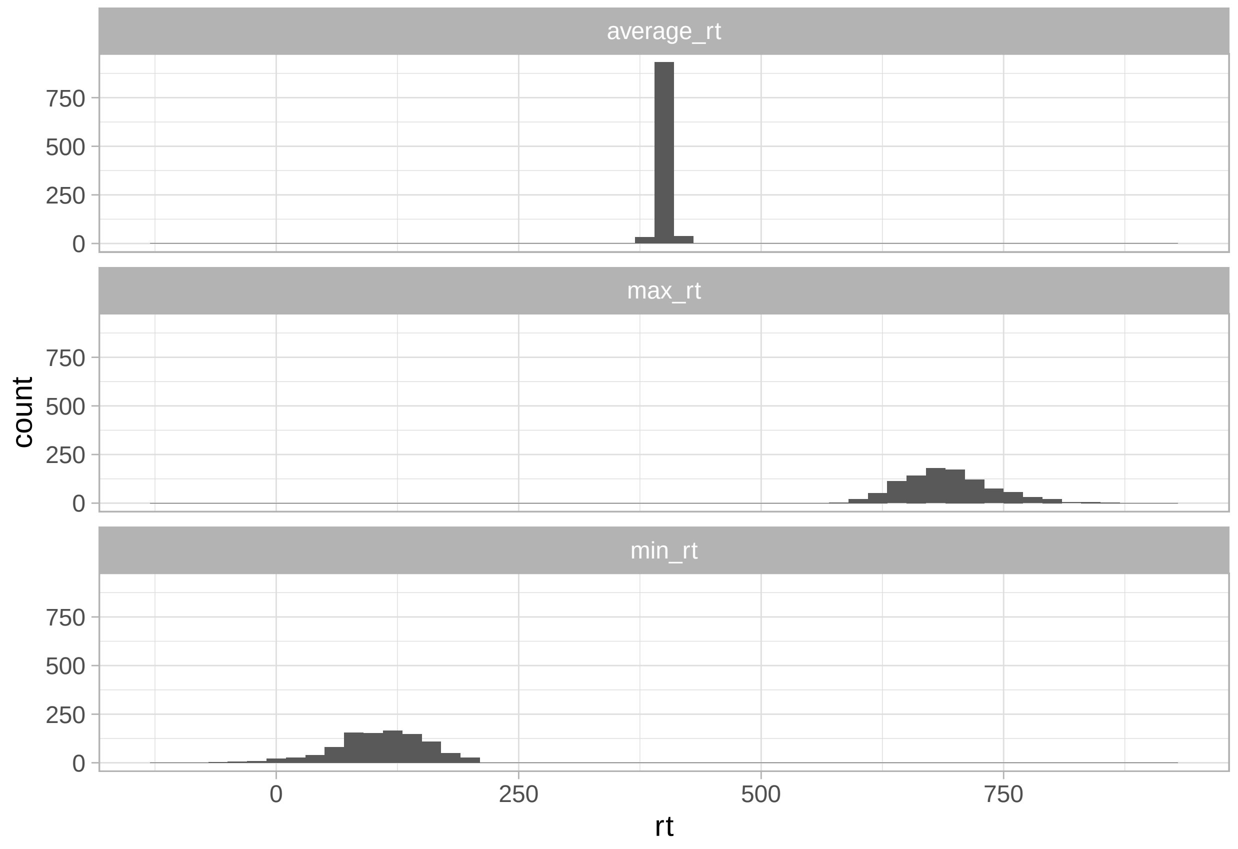


Figure 3: Prior predictive distribution of averages, maximum, and minimum value of the model with too informative priors.

c. Generate posterior predictive distributions based on the model you fit above and plot them.

Option 1: We do the posterior predictive checks using `brms` . We can also use built-in functions to compare more easily the posterior predictive distributions with the observed distribution of response times:

```
pp_check(fit_press_too_inf, ndraws = 100)
```

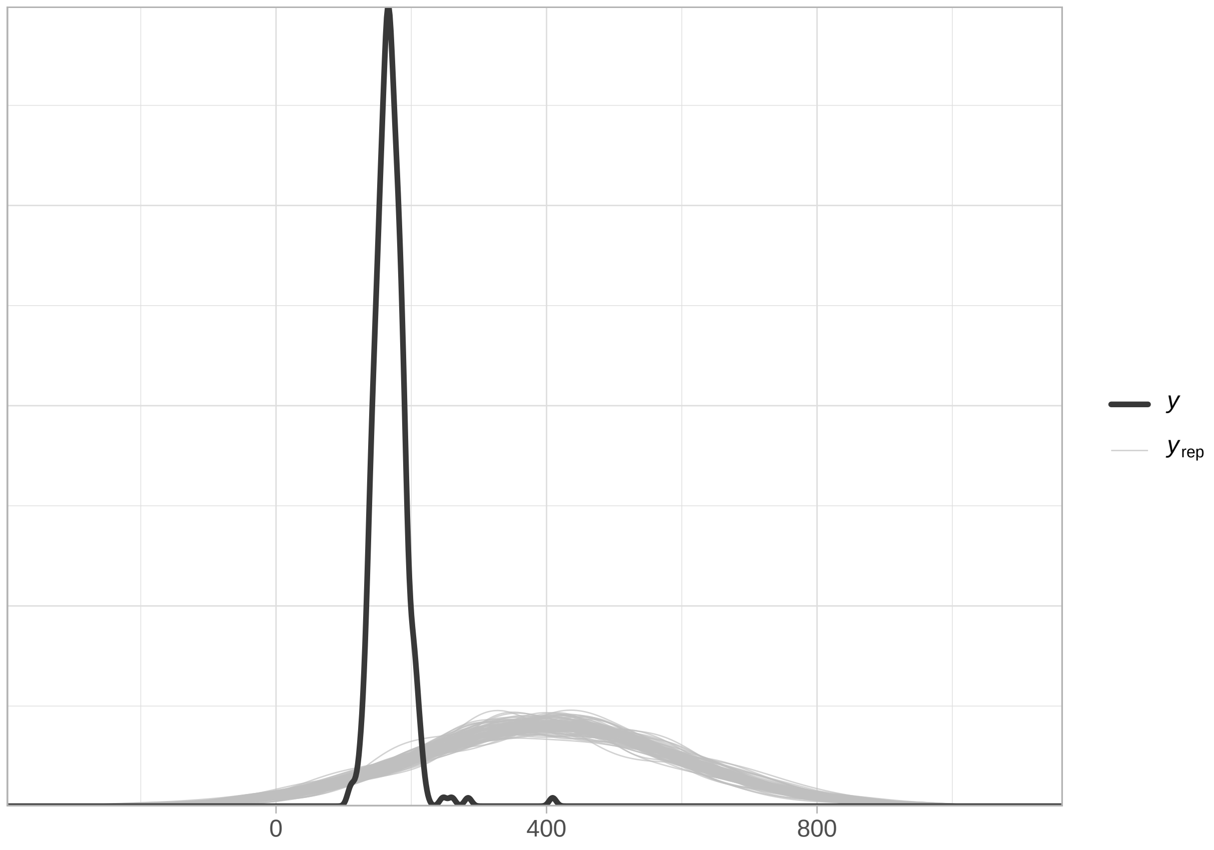


Figure 4: Posterior predictive checks using pp_check.

We see a very clear misfit.

Option 2: One can carry out the posterior predictive checks "by hand".

One can use pretty much the same code as in the book, but the samples are generated from the posteriors rather than from the priors:

```
N_obs <- nrow(df_spacebar)
mu_samples <- as_draws_df(fit_press_too_inf)$b_Intercept
sigma_samples <- as_draws_df(fit_press_too_inf)$sigma
posterior_pred_too_inf <- normal_predictive_distribution_fast(
  mu_samples = mu_samples,
  sigma_samples = sigma_samples,
  N_obs
)
```

```
posterior_pred_too_inf %>%
  filter(iter <= 18) %>%
  ggplot(aes(rt_pred)) +
```

```r
  geom_histogram() +
  facet_wrap(~iter, ncol = 3)
```

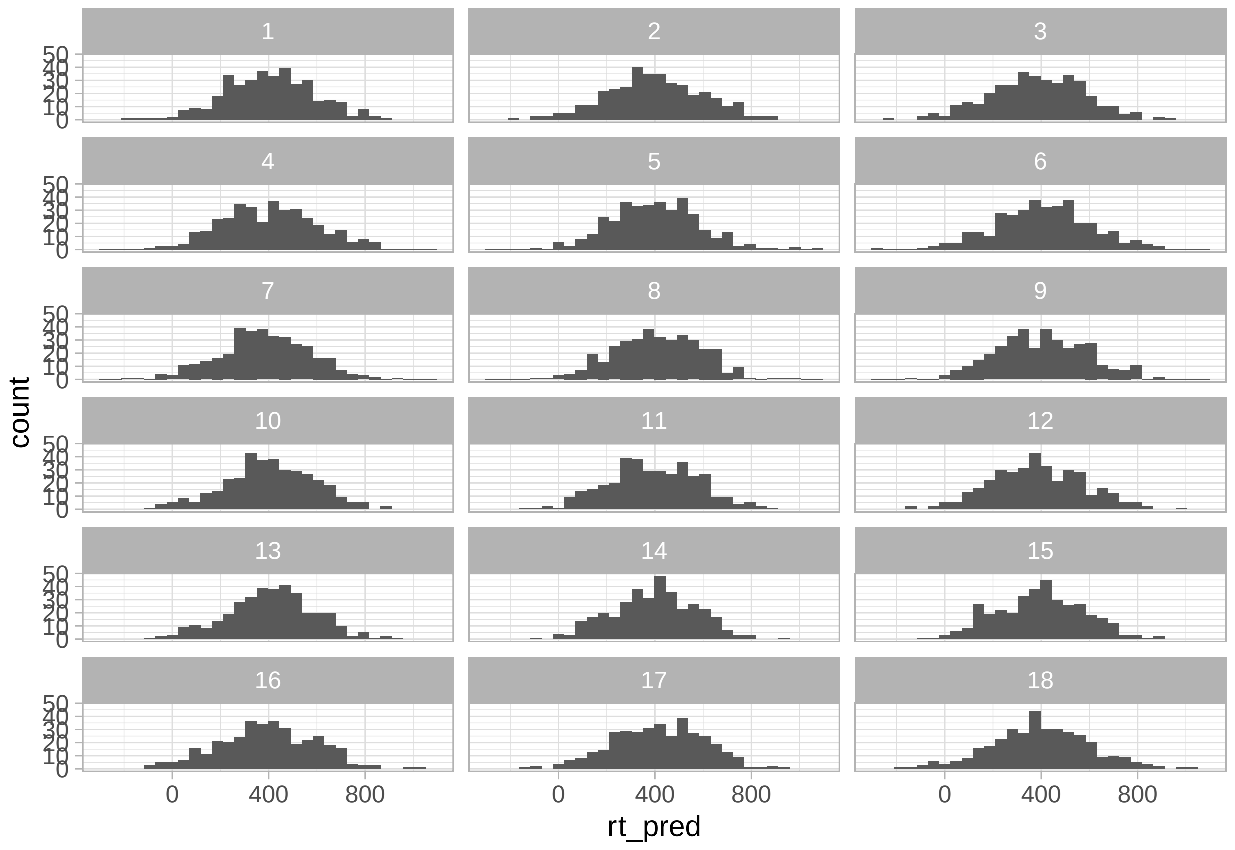


Figure 5: Posterior predictive checks by hand

Both the options above for generating posterior predictive distributions show that these posterior predictive distributions can also be affected by the priors (and not only the likelihood). In general, a selection of priors that noticeably affects the posterior will also affect the posterior predictive distribution.

Exercise 4

(a) For the log-normal model `fit_press_ln`, change the prior of σ so that it is a log-normal distribution with location (μ) of -2 and scale (σ) of $.5$. What does such a prior imply about your belief regarding button-pressing times in milliseconds? Is it a good prior? Generate and plot prior predictive distributions. Do the new estimates change compared to earlier models when you fit the model?

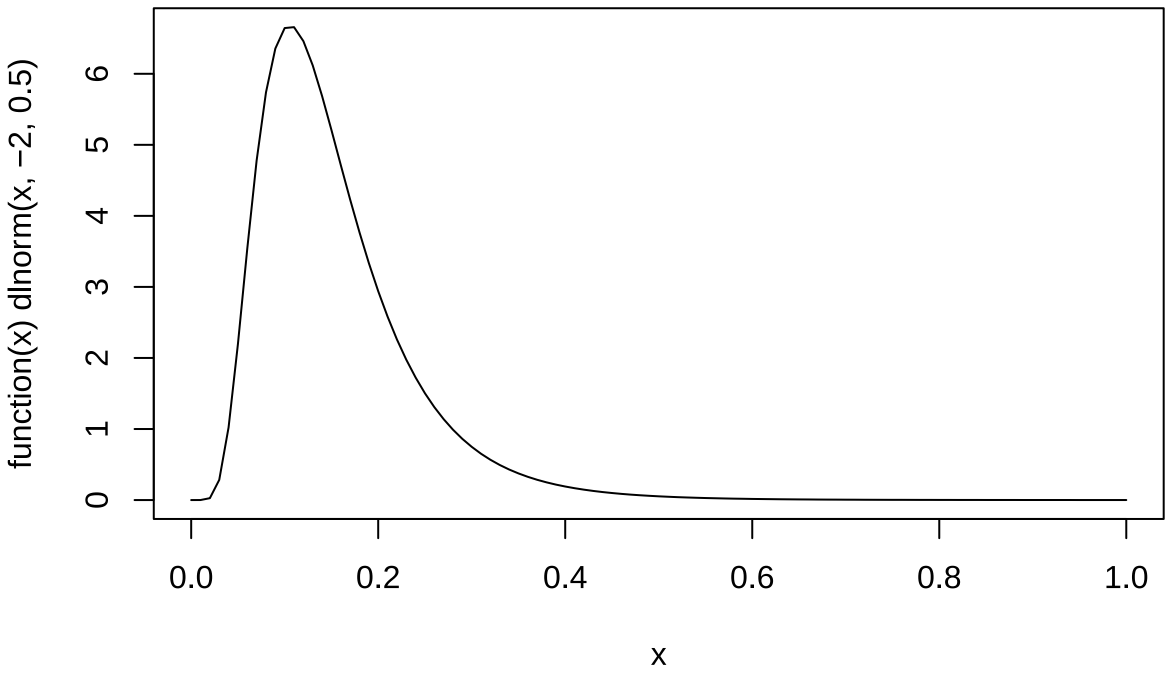
Our model is now:

$$\begin{aligned} rt_n &\sim LogNormal(\mu, \sigma) \\ \mu &\sim Normal(6, 1.5) \\ \sigma &\sim LogNormal(-2, .5) \end{aligned} \tag{2}$$

Note: We don't change our assumptions about the likelihood, that is shape of the distribution of the data, only about our beliefs of the plausible spread of the distribution of the data.

Recall that the parameters of the log-normal distribution are in a different scale than the output of the distribution. This new prior assumes a distribution of possible values of σ as follows:

```r
# Base plot can be useful for a quick and dirty plot here:
plot(function(x) dlnorm(x, -2, 0.5))
```



Calculate the median, mean, and 95% quantiles of the prior distribution of σ by simulation (it is also possible to do it analytically):

```r
quantile(rlnorm(100000, -2, .5), c(.025, .975))
```

```
##   2.5%  97.5%
## 0.0507 0.3615
```

```r
## analytically:
qlnorm(c(0.025,0.975),meanlog = -2,sdlog = 0.5)
```

```
## [1] 0.0508 0.3606
```

```r
median(rlnorm(100000, -2, .5))
```

```
## [1] 0.135
```

```r
## analytically:
qlnorm(0.5,,meanlog = -2,sdlog = 0.5)
```

```
## [1] 0.135
```

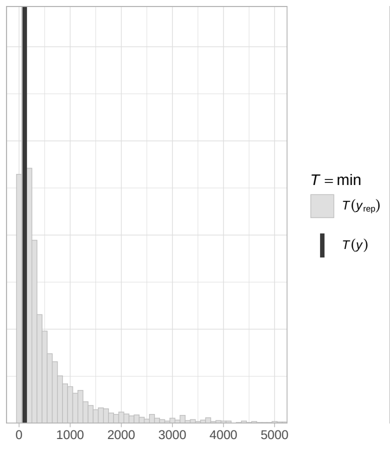
```r
mean(rlnorm(100000, -2, .5))
```

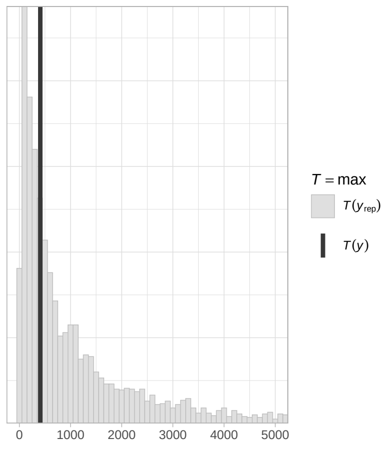
```
## [1] 0.153
```

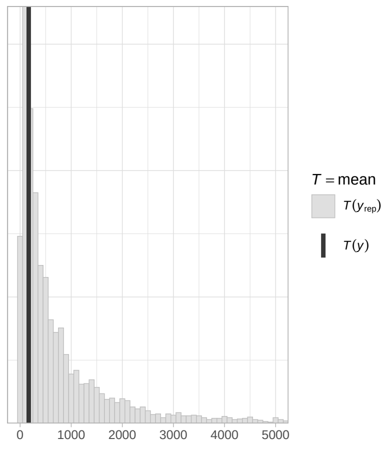
It looks that this prior would actually work quite well. This is also evident from the prior predictive distributions of some representative statistics, which include reasonable values:

```r
fit_press_ln_new_prior <- brm(t ~ 1,
  data = df_spacebar,
  family = lognormal(),
  prior = c(
    prior(normal(6, 1.5), class = Intercept),
    prior(lognormal(-2, .5), class = sigma)
  ),
  sample_prior = "only"
)
```

```r
# We need to adjust the binwidth and the displayed values to get nice plots
pp_check(fit_press_ln_new_prior,
  type = "stat",
  stat = "min",
  binwidth = 100
) +
  coord_cartesian(xlim = c(0, 5000))
pp_check(fit_press_ln_new_prior,
  type = "stat",
  stat = "max",
  binwidth = 100
) +
  coord_cartesian(xlim = c(0, 5000))
pp_check(fit_press_ln_new_prior,
  type = "stat",
  stat = "mean",
  binwidth = 100
) +
  coord_cartesian(xlim = c(0, 5000))
```







Next, fit the model:

```r
fit_press_ln_new <- brm(t ~ 1,
  data = df_spacebar,
  family = lognormal(),
  prior = c(
    prior(normal(6, 1.5), class = Intercept),
    prior(lognormal(-2, .5), class = sigma)
  )
)
```

```r
fit_press_ln_new
```

```
##  Family: lognormal
##   Links: mu = identity; sigma = identity
## Formula: t ~ 1
##    Data: df_spacebar (Number of observations: 361)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Regression Coefficients:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     5.12      0.01     5.10     5.13 1.00     3688     2394
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.13      0.01     0.13     0.14 1.00     3085     2388
##
## Draws were sampled 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).
```

The estimates remain virtually the same as before.

(c) For the log-normal model, what is the mean (rather than median) time that takes to press the space bar, what is the standard deviation of the response times in milliseconds To work out how long it takes to press the space bar on average (rather than the median) in milliseconds, transform μ using the following formula: $\exp(\mu + \sigma^2/2)$ (it doesn't really matter whether one uses `fit_press_ln` or `fit_press_ln_new`, given that the estimates were so close to each other):

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

```r
mu <- as_draws_df(fit_press_ln)$b_Intercept
sigma <- as_draws_df(fit_press_ln)$sigma
mean_estimate_ms <- exp(mu + (sigma^2) / 2)
```

The above yields 4000 samples of the posterior mean RT, whch can be summarized as follows:

```r
c(mean = mean(mean_estimate_ms), quantile(mean_estimate_ms, probs = c(.025, .975)))
```

```
##  mean  2.5% 97.5%
##   169   166   171
```

For estimating the standard deviation of the response times, use the following formula: $\exp(\mu + \sigma^2/2) \times \sqrt(\exp(\sigma^2) - 1)$:

```r
sd_estimate_ms <- exp(mu + (sigma^2) / 2) * sqrt(exp(sigma^2) - 1)
```

Summarizing the results:

```r
c(mean = mean(sd_estimate_ms), quantile(sd_estimate_ms, probs = c(.025, .975)))
```

```
##  mean  2.5% 97.5%
##  22.8  21.2  24.7
```

Exercise 5

a. Fit this model with a prior that assigns approximately 95% of the prior probability of `alpha` to be between 0 and 10.

We need a prior for α as well now. We use a prior that assigns approximately 95% of the prior probability to be between 0 and 10. If we assume a normal distribution centered in 5, with a standard deviation of 2.5.

```
quantile(rnorm(100000, 5, 2.5), c(.025, .975))
```

```
##   2.5%  97.5%
## 0.0904 9.9229
```

```
## using a built-in function:
qnorm(c(0.025,0.975),mean=5,sd=2.5)
```

```
## [1] 0.1 9.9
```

We use this prior for α in the following model:

```
fit_press_skew <- brm(t ~ 1,
  data = df_spacebar,
  family = skew_normal(),
  prior = c(
    # for the first two parameters, I just decided on these priors
    # that look sensible to me
    prior(normal(200, 100), class = Intercept),
    prior(normal(50, 25), class = sigma),
    prior(normal(5, 2.5), class = alpha)
  )
)
```

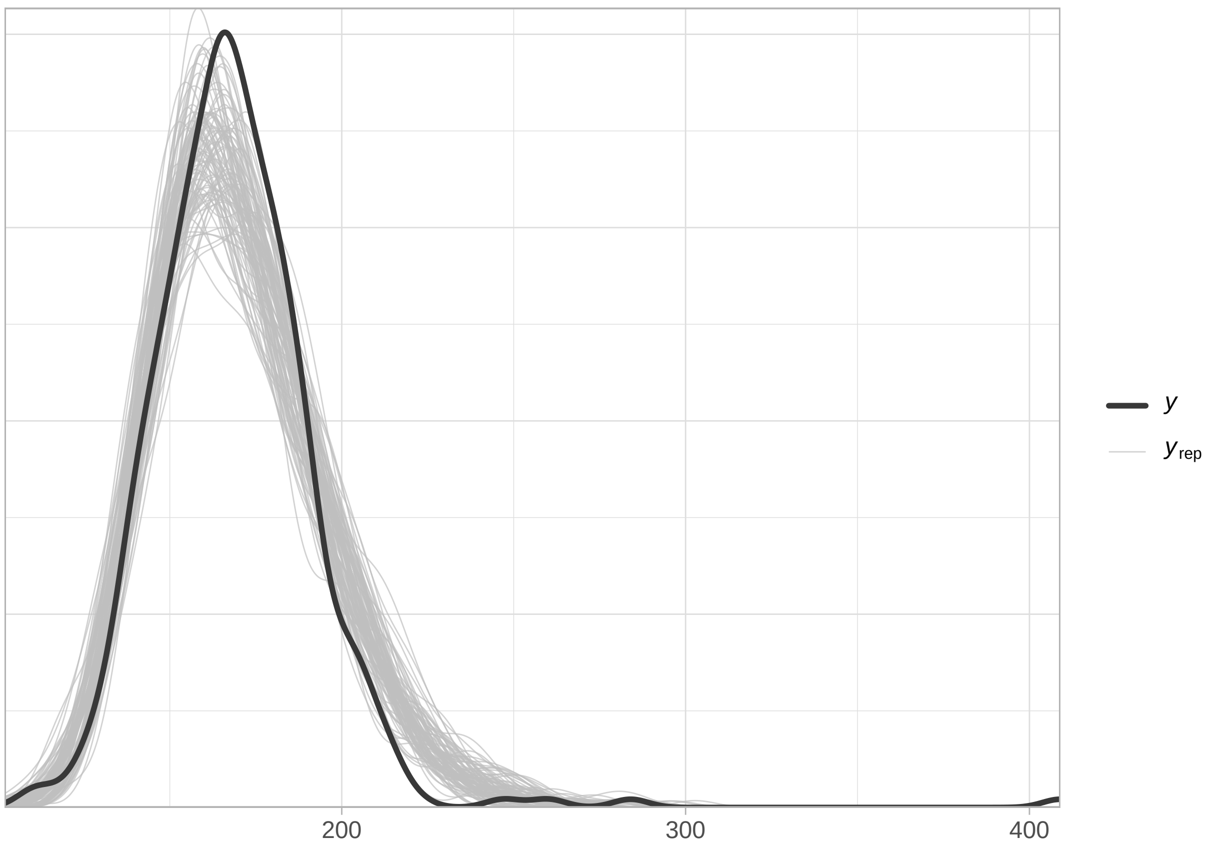
```
fit_press_skew
```

```
##  Family: skew_normal
##   Links: mu = identity; sigma = identity; alpha = identity
## Formula: t ~ 1
##    Data: df_spacebar (Number of observations: 361)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Regression Coefficients:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept   169.70      1.26   167.24   172.23 1.00     2754     2655
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    24.11      0.96    22.29    26.07 1.00     2405     2589
## alpha     2.67      0.41     1.96     3.53 1.00     2560     2148
##
## Draws were sampled 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).
```

b. Generate posterior predictive distributions and compare the posterior distribution of summary statistics of the skew normal with the normal and log-normal

Posterior predictive distributions can show us the fit of this new model:

```r
pp_check(fit_press_skew, ndraws = 100)
```



It's not better than the fit of the "regular" normal distribution. We examine the distribution of minimum and maximum values below and we compare them with models that assume a "regular" normal distribution and log-normal distribution as likelihood functions.

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

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

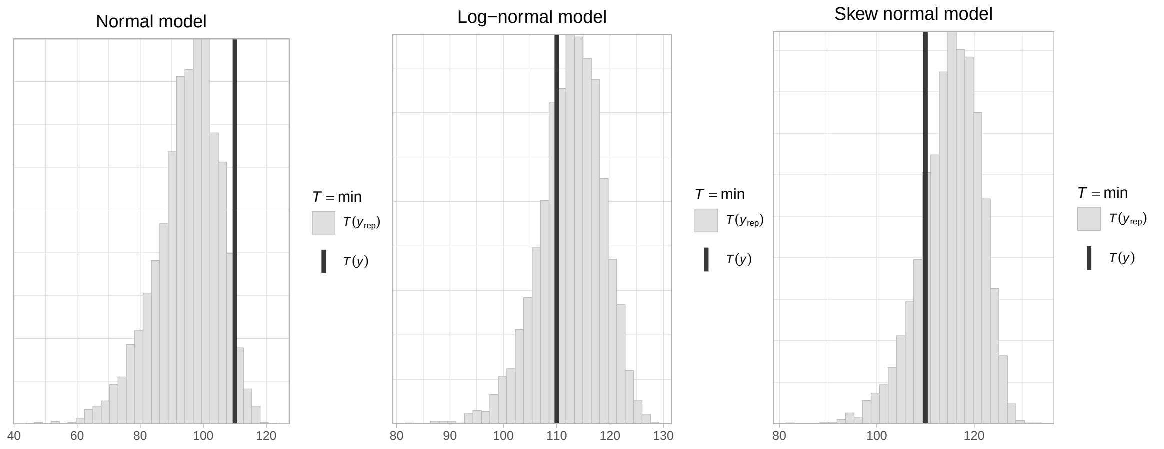


Figure 6: Distribution of minimum values in a posterior predictive check.

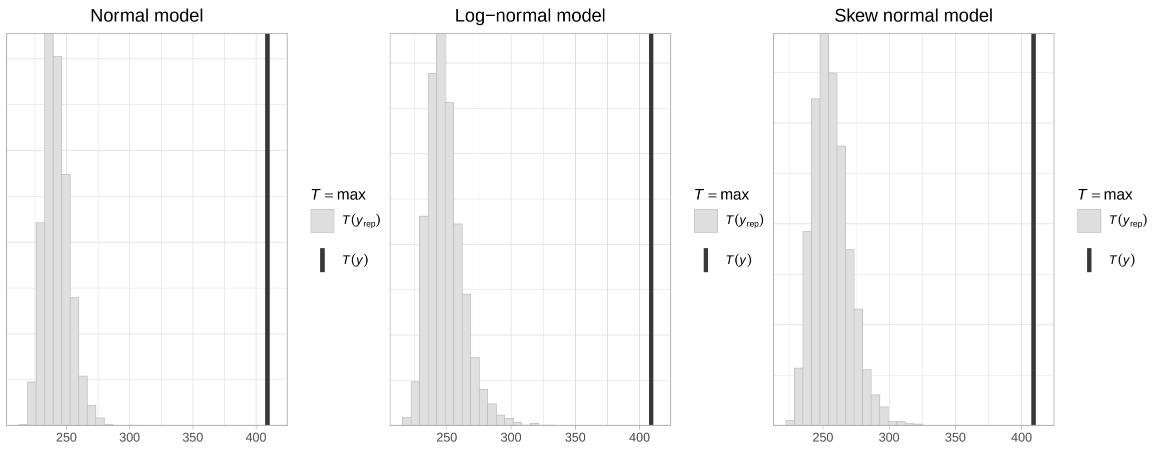


Figure 7: Distribution of maximum values in a posterior predictive check.