

Prior Predictive Checks: Reaction Time Example

Bayesian Mixed Effects Models with brms for Linguists

Workshop Materials

2025-11-11

Table of contents

1	Prior Predictive Checks for Reaction Time Data	1
1.1	Why Prior Predictive Checks Matter	2
1.2	Setup	2
1.3	Fitting Prior Only Model	3
1.4	Prior Distributions	8
1.4.1	Intercept Prior	8
1.4.2	Effect Size Prior	10
1.4.3	Residual Noise Prior	11
1.5	Random Effects Distributions	11
1.5.1	Subject Random Intercepts	11
1.5.2	Subject Random Slopes	13
1.5.3	Residual Noise Distribution	15
1.5.4	Random Effects Comparison	16
1.6	Interpretation	17
1.6.1	Good Signs (Prior is Reasonable)	17
1.6.2	Problems to Watch For	17
1.7	Summary	17

1 Prior Predictive Checks for Reaction Time Data

This document demonstrates how to validate priors for a Bayesian RT model before fitting to data.

1.1 Why Prior Predictive Checks Matter

Before fitting your model to actual data, validate that your priors generate sensible predictions. This is crucial because: - A prior that's too restrictive can prevent the model from learning from data - A prior that's too permissive might not regularize your estimates - It's much easier to revise priors before fitting than after

1.2 Setup

```
library(brms)
library(tidyverse)
library(bayesplot)
library(posterior) # For as_draws_df() - following Kurz's approach

# Create example RT data
set.seed(42)
n_subj <- 20
n_trials <- 50
n_items <- 30

rt_data <- expand_grid(
  trial = 1:n_trials,
  subject = 1:n_subj,
  item = 1:n_items
) %>%
  filter(row_number() <= n_subj * n_trials * 3) %>%
  mutate(
    condition = rep(c("A", "B"), length.out = n()),
    log_rt = rnorm(n(), mean = 6, sd = 0.3) +
      (condition == "B") * 0.15 +
      rnorm(n(), mean = 0, sd = 0.1),
    rt = exp(log_rt)
  )

# Define priors
rt_priors <- c(
  prior(normal(6, 1.5), class = Intercept),
  prior(normal(0, 0.5), class = b),
  prior(exponential(1), class = sigma),
  prior(exponential(1), class = sd),
```

```
prior(lkj(2), class = cor)
)
```

1.3 Fitting Prior Only Model

```
# Create fits directory if it doesn't exist
if (!dir.exists("fits")) dir.create("fits")

# Fit model with priors only
# Using file argument to cache the fitted model and speed up re-runs
prior_pred <- brm(
  log_rt ~ condition + (1 + condition | subject) + (1 | item),
  data = rt_data,
  family = gaussian(),
  prior = rt_priors,
  sample_prior = "only",
  chains = 4, iter = 1000, # 4 chains, fewer iterations for prior checks
  cores = 4,               # Use 4 cores for parallel sampling
  verbose = FALSE,
  refresh = 0,
  file = "fits/prior_pred_rt", # Cache model - only refits if data/formula/priors change
  file_refit = "on_change"     # Only refit when necessary
)
```

```
## Prior Predictive Checks
```

```
### Visual Checks
```

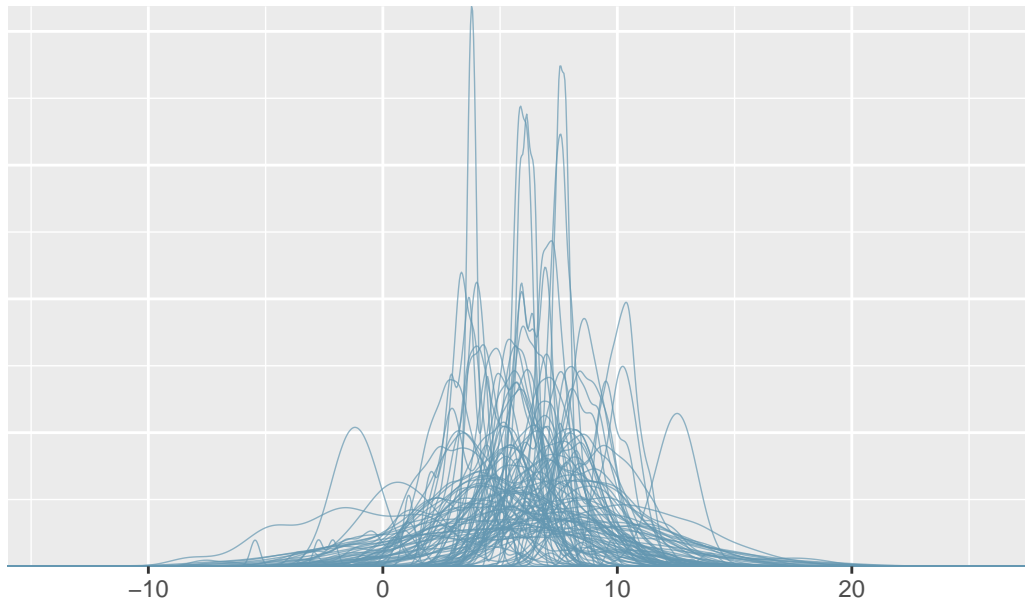
```
::: {.cell}
```

```
```{r .cell-code}
```

```
Density overlay
```

```
pp_check(prior_pred, type = "dens_overlay", ndraws = 100, prefix = "ppd") +
 labs(title = "Prior Predictions vs Observed Data (Density Overlay)")
```

## Prior Predictions vs Observed Data (Density Overlay)



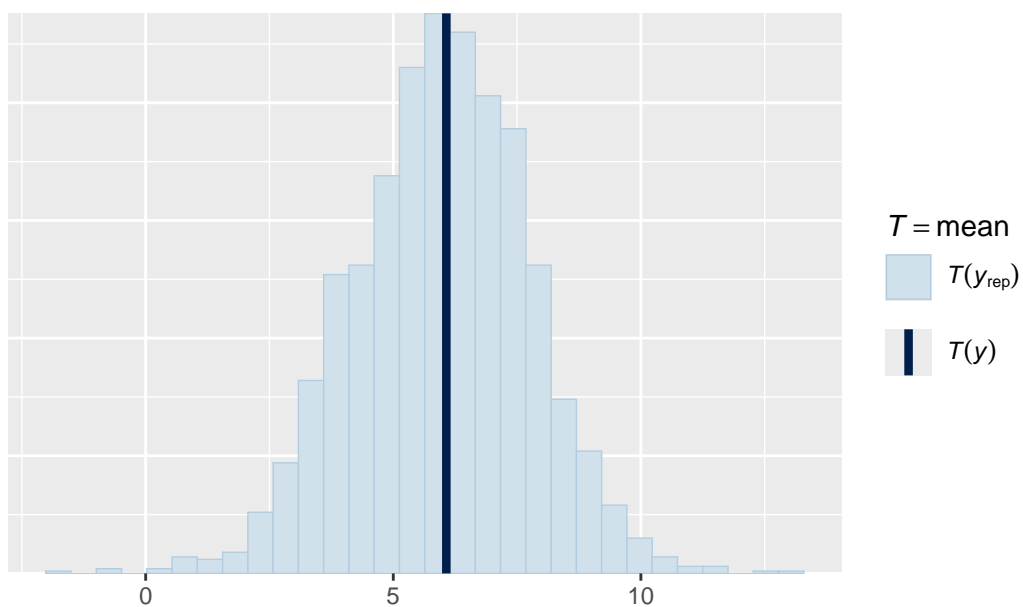
```
Mean comparison
pp_check(prior_pred, type = "stat", stat = "mean") +
 labs(title = "Prior Predictions vs Observed Data (Mean)")
```

Using all posterior draws for ppc type 'stat' by default.

Note: in most cases the default test statistic 'mean' is too weak to detect anything of interest.

Use `pp_check(prior_pred, type = "stat", stat = "mean", bins = 30)` using `bins = 30`. Pick better value `binwidth`.

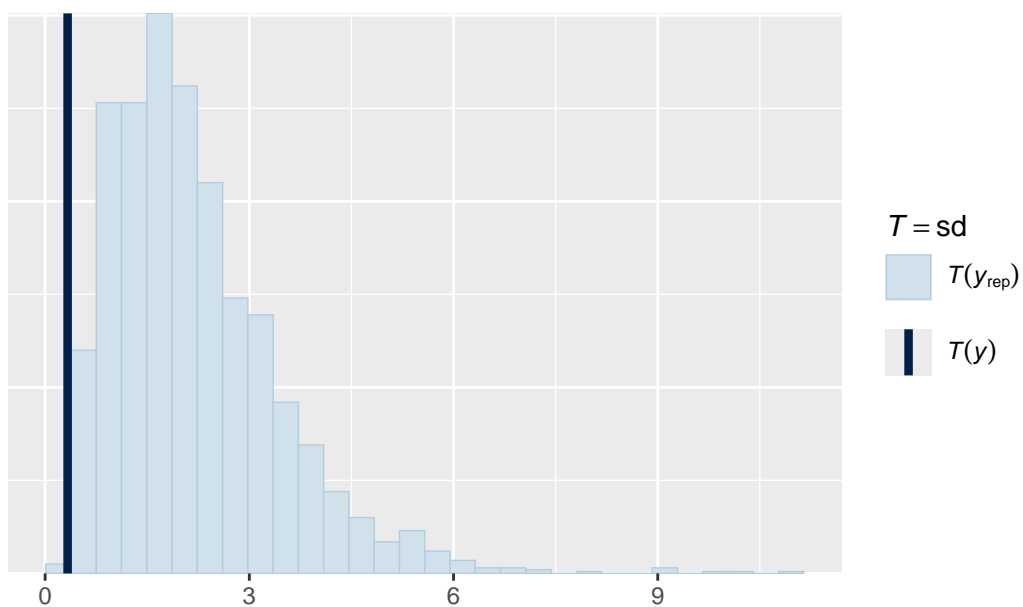
## Prior Predictions vs Observed Data (Mean)



```
SD comparison
pp_check(prior_pred, type = "stat", stat = "sd") +
 labs(title = "Prior Predictions vs Observed Data (SD)")
```

Using all posterior draws for ppc type 'stat' by default.  
`stat\_bin()` using `bins = 30`. Pick better value `binwidth`.

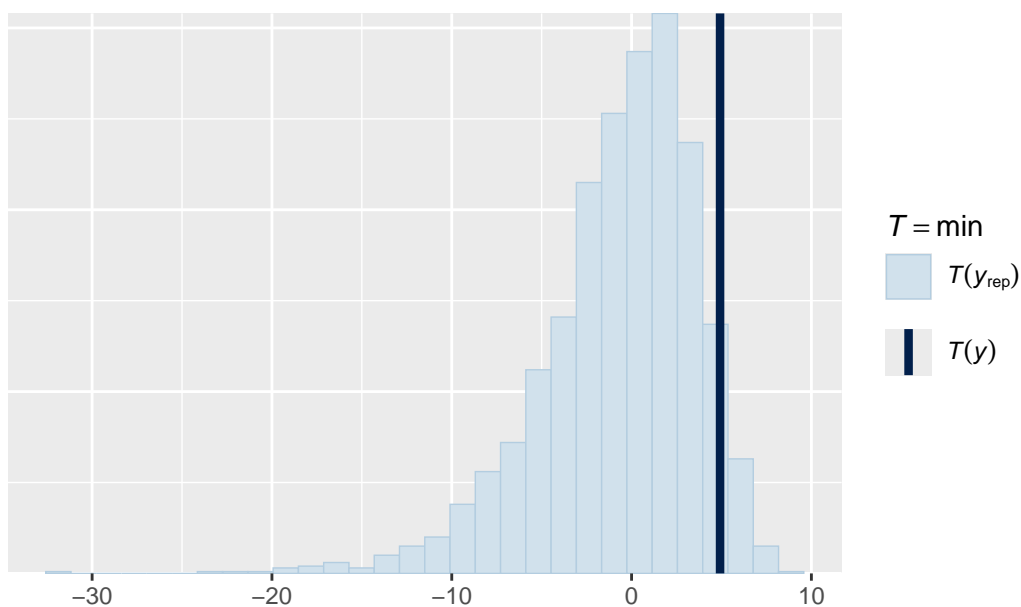
## Prior Predictions vs Observed Data (SD)



```
Min comparison
pp_check(prior_pred, type = "stat", stat = "min") +
 labs(title = "Prior Predictions vs Observed Data (Min)")
```

Using all posterior draws for ppc type 'stat' by default.  
`stat\_bin()` using `bins = 30`. Pick better value `binwidth`.

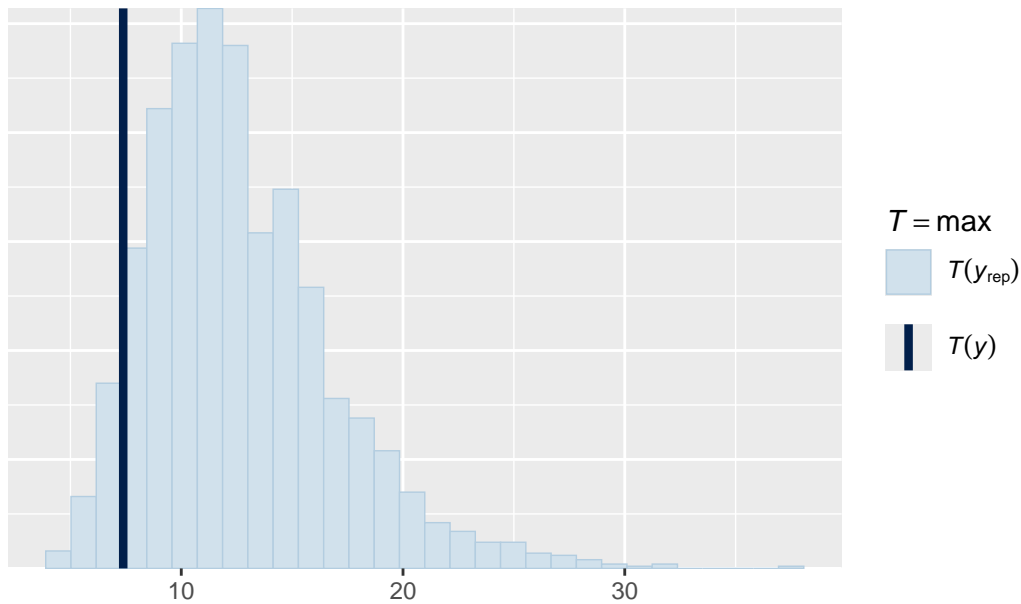
## Prior Predictions vs Observed Data (Min)



```
Max comparison
pp_check(prior_pred, type = "stat", stat = "max") +
 labs(title = "Prior Predictions vs Observed Data (Max)")
```

Using all posterior draws for ppc type 'stat' by default.  
`stat\_bin()` using `bins = 30`. Pick better value `binwidth`.

## Prior Predictions vs Observed Data (Max)



...

### 1.4 Prior Distributions

Following Kurz's approach, we extract prior samples using `as_draws_df()` from the posterior package.

```
Extract prior draws as a data frame (Kurz's method)
This works seamlessly with tidyverse and gives us a proper data frame
library(posterior)
prior_samples <- as_draws_df(prior_pred)
```

#### 1.4.1 Intercept Prior

```
Now we can access columns directly from the data frame
intercept_vals <- prior_samples$b_Intercept

cat("Intercept prior (log scale):\n")
```

Intercept prior (log scale):



```
print(quantile(intercept_vals, c(0.025, 0.5, 0.975), na.rm = TRUE))
```

```
 2.5% 50% 97.5%
2.875128 6.020023 8.923694
```

```
cat("\nIntercept prior (RT scale in milliseconds):\n")
```

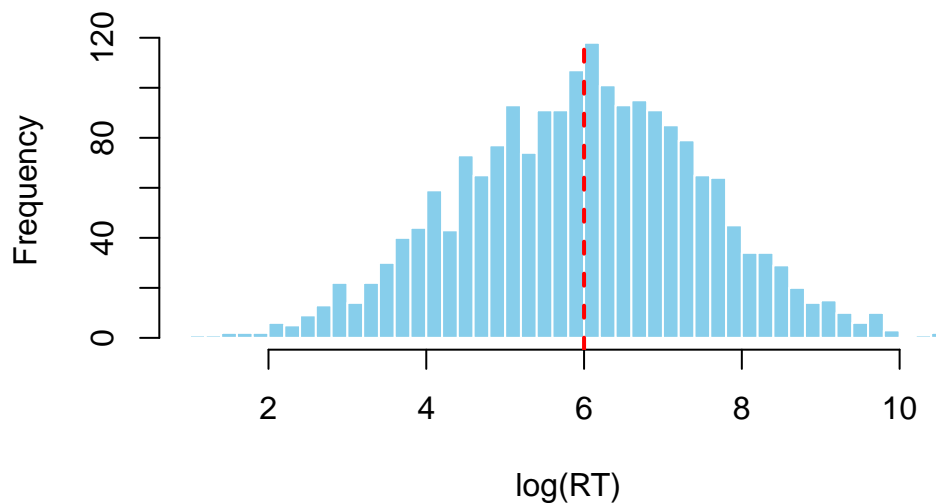
```
Intercept prior (RT scale in milliseconds):
```

```
print(exp(quantile(intercept_vals, c(0.025, 0.5, 0.975), na.rm = TRUE)))
```

```
 2.5% 50% 97.5%
17.72769 411.58799 7507.77385
```

```
Visualization
hist(intercept_vals,
 main = "Prior for Intercept (log-RT scale)",
 xlab = "log(RT)",
 breaks = 50, col = "skyblue", border = "white")
abline(v = 6, col = "red", lwd = 2, lty = 2)
```

### Prior for Intercept (log-RT scale)



### 1.4.2 Effect Size Prior

```
effect_vals <- prior_samples$b_conditionB

cat("Condition effect prior (log scale):\n")
```

Condition effect prior (log scale):

```
effect_q <- quantile(effect_vals, c(0.025, 0.5, 0.975), na.rm = TRUE)
print(effect_q)
```

```
 2.5% 50% 97.5%
-0.99889479 -0.01414723 0.94791795
```

```
cat("\nCondition effect prior (RT scale in ms):\n")
```

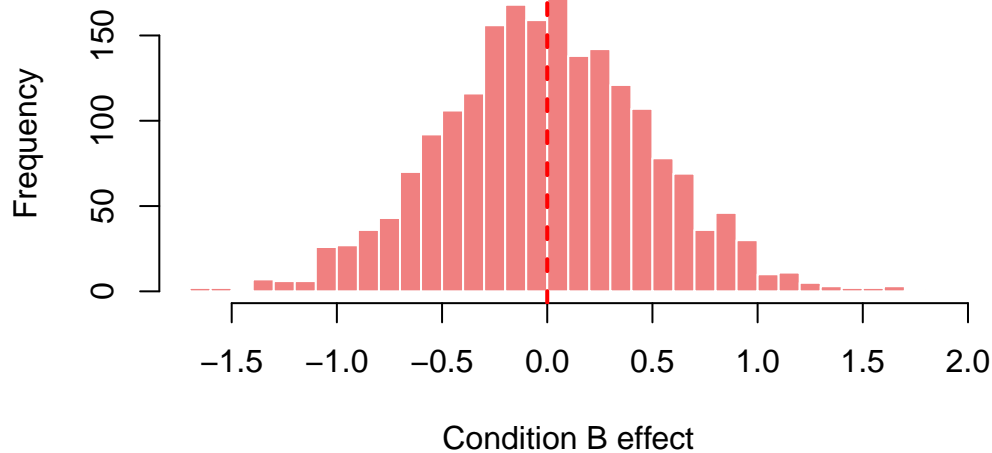
Condition effect prior (RT scale in ms):

```
print(exp(effect_q))
```

```
 2.5% 50% 97.5%
0.3682863 0.9859524 2.5803317
```

```
Visualization
hist(effect_vals,
 main = "Prior for Condition Effect (log-RT scale)",
 xlab = "Condition B effect",
 breaks = 50, col = "lightcoral", border = "white")
abline(v = 0, col = "red", lwd = 2, lty = 2)
```

## Prior for Condition Effect (log-RT scale)



### 1.4.3 Residual Noise Prior

```
sigma_vals <- prior_samples$sigma
cat("Residual noise prior (sigma, log scale):\n")
```

Residual noise prior (sigma, log scale):

```
sigma_q <- quantile(sigma_vals, c(0.025, 0.5, 0.975), na.rm = TRUE)
print(sigma_q)
```

2.5%	50%	97.5%
0.02573465	0.69876976	3.81711126

## 1.5 Random Effects Distributions

For prior predictive checks, we examine the **implied distribution** of subject-specific parameters by extracting the hyperprior SDs and simulating random effects.

### 1.5.1 Subject Random Intercepts

```
Extract hyperprior SDs from prior samples
sd_subject_intercept <- prior_samples$sd_subject__Intercept

cat("Subject random intercept SD prior:\n")
```

Subject random intercept SD prior:

```
print(quantile(sd_subject_intercept, c(0.025, 0.5, 0.975)))
```

```
 2.5% 50% 97.5%
0.02519548 0.67840017 3.61983168
```

```
Simulate random effects from this prior
set.seed(123)
n_sims <- 1000
simulated_intercepts <- rnorm(n_sims, mean = 0, sd = median(sd_subject_intercept))

cat("\nImplied subject random intercepts (log scale):\n")
```

Implied subject random intercepts (log scale):

```
subject_int <- quantile(simulated_intercepts, c(0.025, 0.5, 0.975))
print(subject_int)
```

```
 2.5% 50% 97.5%
-1.317150788 0.006247821 1.382502744
```

```
cat("\nImplied subject-specific RTs (milliseconds):\n")
```

Implied subject-specific RTs (milliseconds):

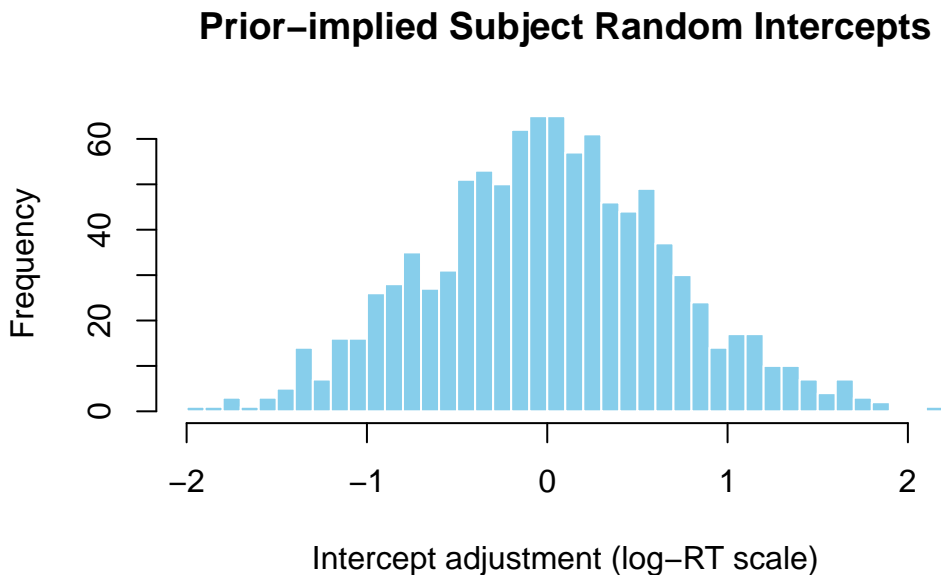
```
Combine with typical intercept value from prior
intercept_median <- median(intercept_vals)
subject_rt <- exp(intercept_median + subject_int)
print(subject_rt)
```

2.5%      50%      97.5%  
110.2634   414.1676   1640.1214

```
cat("Prior implies subject RTs range from",
 round(subject_rt[1]), "to", round(subject_rt[3]), "ms\n")
```

Prior implies subject RTs range from 110 to 1640 ms

```
Visualization
hist(simulated_intercepts,
 main = "Prior-implied Subject Random Intercepts",
 xlab = "Intercept adjustment (log-RT scale)",
 breaks = 30, col = "skyblue", border = "white")
```



### 1.5.2 Subject Random Slopes

```
Extract hyperprior SDs for slopes
sd_subject_slope <- prior_samples$sd_subject__conditionB

cat("Subject random slope SD prior:\n")
```

Subject random slope SD prior:

```
print(quantile(sd_subject_slope, c(0.025, 0.5, 0.975)))
```

```
 2.5% 50% 97.5%
0.0249993 0.6998014 3.6529919
```

```
Simulate random slope effects
simulated_slopes <- rnorm(n_sims, mean = 0, sd = median(sd_subject_slope))

cat("\nImplied subject random slopes (log scale):\n")
```

Implied subject random slopes (log scale):

```
subject_slope <- quantile(simulated_slopes, c(0.025, 0.5, 0.975))
print(subject_slope)
```

```
 2.5% 50% 97.5%
-1.39369961 0.03838577 1.33375954
```

```
cat("\nInterpretation: Condition effect varies by subject\n")
```

Interpretation: Condition effect varies by subject

```
cat("Small effect subjects (2.5%): ", round(exp(subject_slope[1]), 3), "× multiplier\n")
```

Small effect subjects (2.5%): 0.248 × multiplier

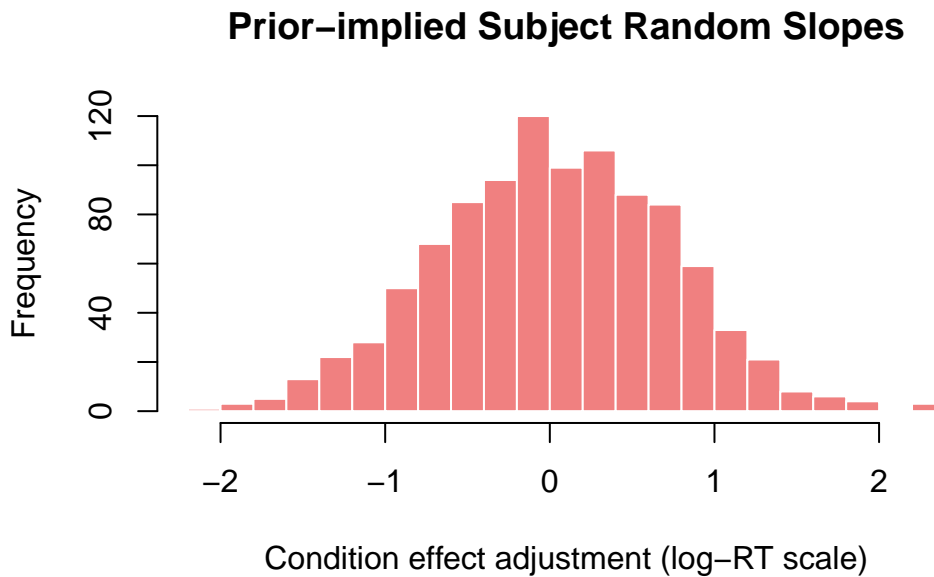
```
cat("Average effect subjects (50%): ", round(exp(subject_slope[2]), 3), "× multiplier\n")
```

Average effect subjects (50%): 1.039 × multiplier

```
cat("Large effect subjects (97.5%): ", round(exp(subject_slope[3]), 3), "× multiplier\n")
```

Large effect subjects (97.5%): 3.795 × multiplier

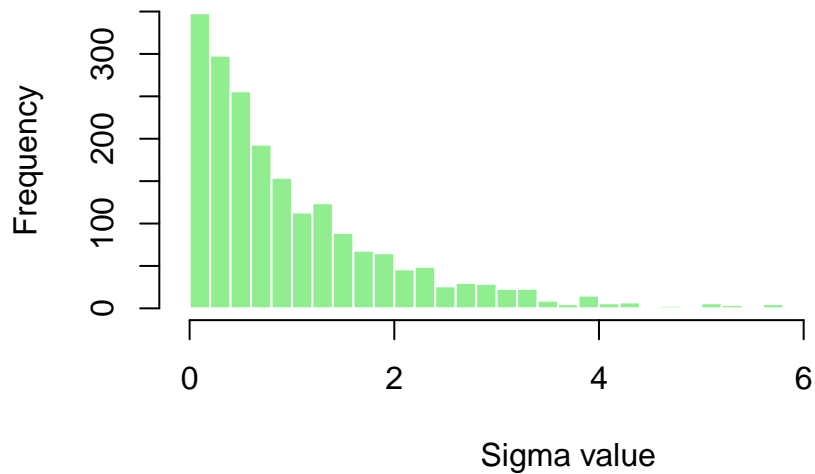
```
Visualization
hist(simulated_slopes,
 main = "Prior-implied Subject Random Slopes",
 xlab = "Condition effect adjustment (log-RT scale)",
 breaks = 30, col = "lightcoral", border = "white")
```



### 1.5.3 Residual Noise Distribution

```
if (!is.null(sigma_vals) && is.numeric(sigma_vals)) {
 hist(sigma_vals,
 main = "Prior for Residual Noise (Sigma)",
 xlab = "Sigma value",
 breaks = 30, col = "lightgreen", border = "white")
} else {
 plot(1, main = "Sigma visualization", xlab = "", ylab = "")
 text(1, 1, "Sigma samples not available")
}
```

## Prior for Residual Noise (Sigma)

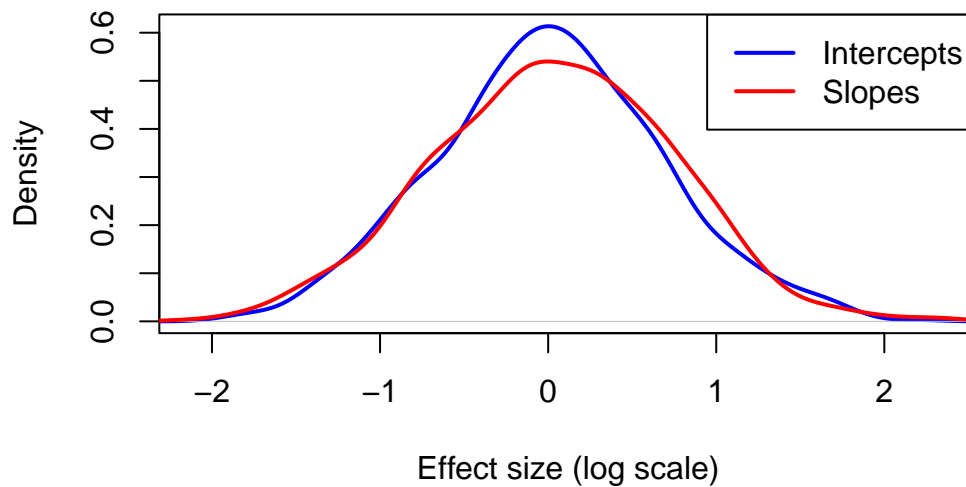


### 1.5.4 Random Effects Comparison

```
Density plot comparing intercepts and slopes
plot(density(simulated_intercepts),
 main = "Prior-implied Random Effects Distributions",
 xlab = "Effect size (log scale)",
 col = "blue", lwd = 2, xlim = range(c(simulated_intercepts, simulated_slopes)),
 ylim = c(0, max(density(simulated_intercepts)$y, density(simulated_slopes)$y)))
lines(density(simulated_slopes),
 col = "red", lwd = 2)
legend("topright", c("Intercepts", "Slopes"), col = c("blue", "red"), lwd = 2)
```



## Prior-implied Random Effects Distributions



### 1.6 Interpretation

#### 1.6.1 Good Signs (Prior is Reasonable)

- Prior generates log-RTs around 6 ( 400ms)
- 95% interval roughly 200-1100ms (plausible RT range)
- Condition effect typically < 150ms difference
- Between-subject variation is moderate

#### 1.6.2 Problems to Watch For

- Mean RT » 1000ms: intercept prior too high
- 95% interval 10ms-50s: priors too wide
- No variation between subjects: SD priors too small

### 1.7 Summary

Before fitting your model to actual data, always validate that your priors: 1. Generate reasonable predictions 2. Allow the data to inform the posterior 3. Respect domain knowledge constraints

Adjust priors as needed and rerun these checks.