Part-1:

```
# Install TinyTeX with force = TRUE
tinytex::install_tinytex(force = TRUE)
```

```
## tlmgr --repository http://www.preining.info/tlgpg/ install tlgpg

## tlmgr option repository "https://in.mirrors.cicku.me/ctan/systems/texlive/tlnet"

## tlmgr update --list
```

```
df_powerpose <- read.csv("df_powerpose.csv")
head(df_powerpose)
```

```
##   X id hptreat female age testm1  testm2
## 1 2 29    High   Male  19 38.725  62.375
## 2 3 30     Low Female  20 32.770  29.235
## 3 4 31    High Female  20 32.320  27.510
## 4 5 32     Low Female  18 17.995  28.655
## 5 7 34     Low Female  21 73.580  44.670
## 6 8 35    High Female  20 80.695 105.485
```

1.1

```
df_powerpose <- df_powerpose %>%
  mutate(hptreat_num = ifelse(hptreat == "High", 1, 0))
df_powerpose <- df_powerpose %>%
  mutate(testosterone_change = testm2 - testm1)
priors <- c(
  prior(normal(0, 10), class = Intercept),
  prior(normal(0, 10), class = b, coef = hptreat_num)
)
mfit <- brm(
  formula = testosterone_change ~ 1 + hptreat_num,
  data = df_powerpose,
  family = gaussian(),
  prior = priors,
  chains = 4,
  cores = 4,
  iter = 2000,
  warmup = 1000
)
```

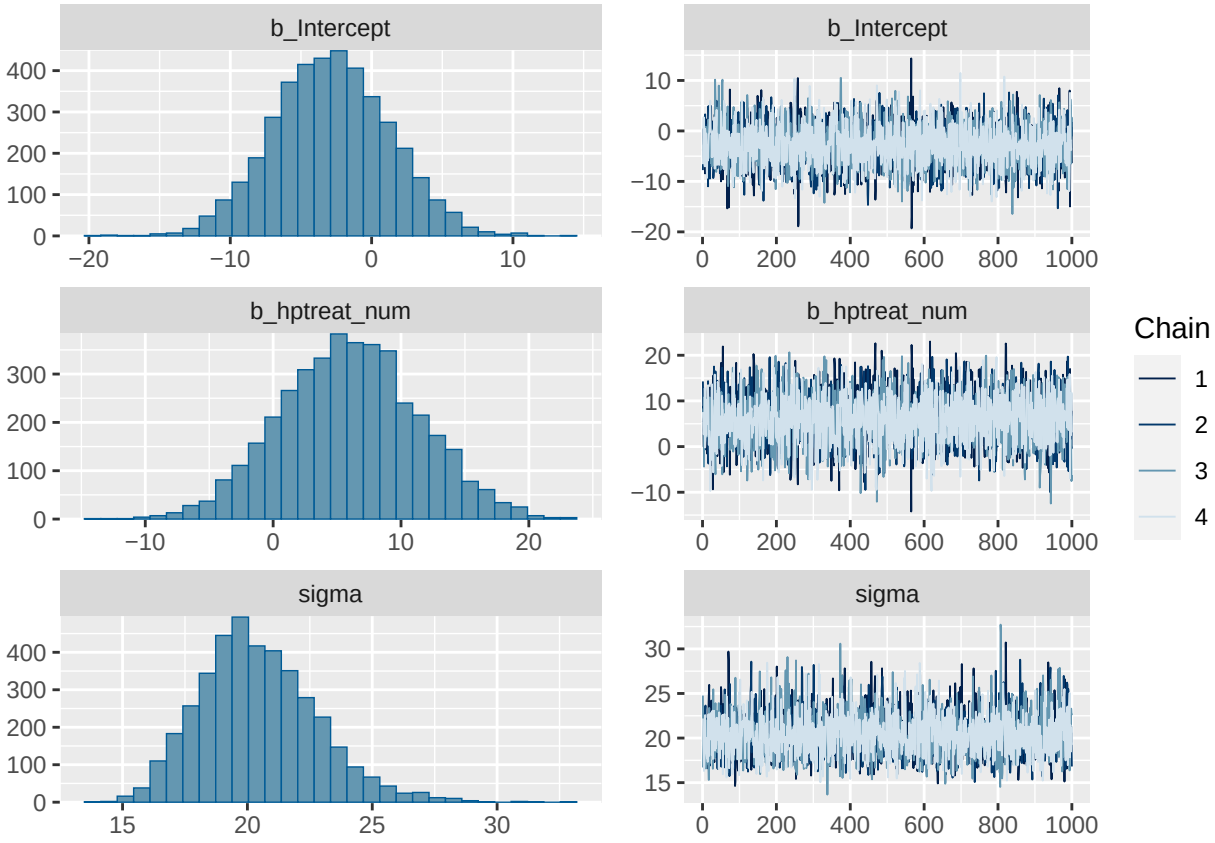
```
## Compiling Stan program...

## Start sampling
```

```
save(mfit, file = "FittedModels/PowerPoseTestosterone.Rda")
summary(mfit)
```

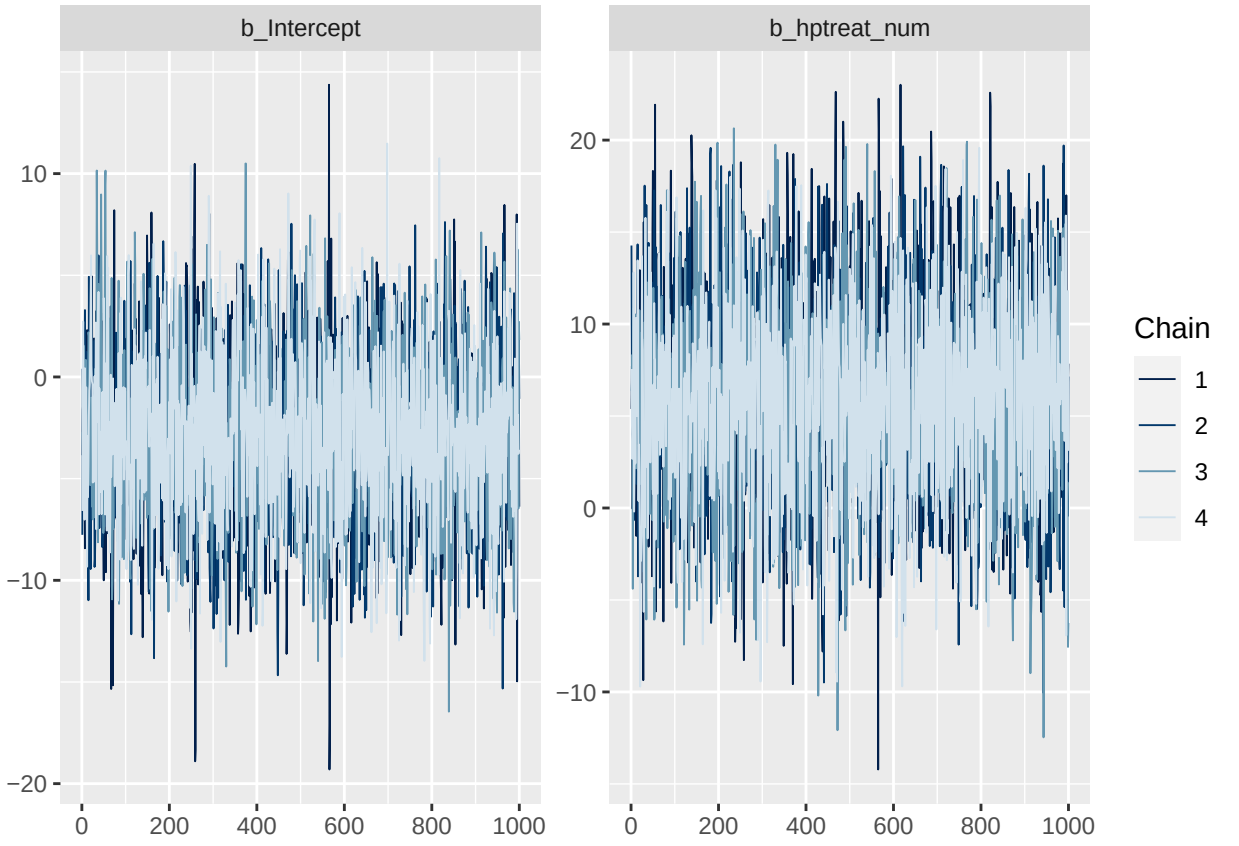
```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: testosterone_change ~ 1 + hptreat_num
##    Data: df_powerpose (Number of observations: 39)
```

```
##   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      -2.85      4.11   -10.70     5.23 1.00     3776     2693
## hptreat_num     6.00      5.41    -4.33    16.73 1.00     3807     2718
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    20.44      2.36    16.42    25.61 1.00     3116     2354
##
## 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).
```

```
plot(mfit)
```



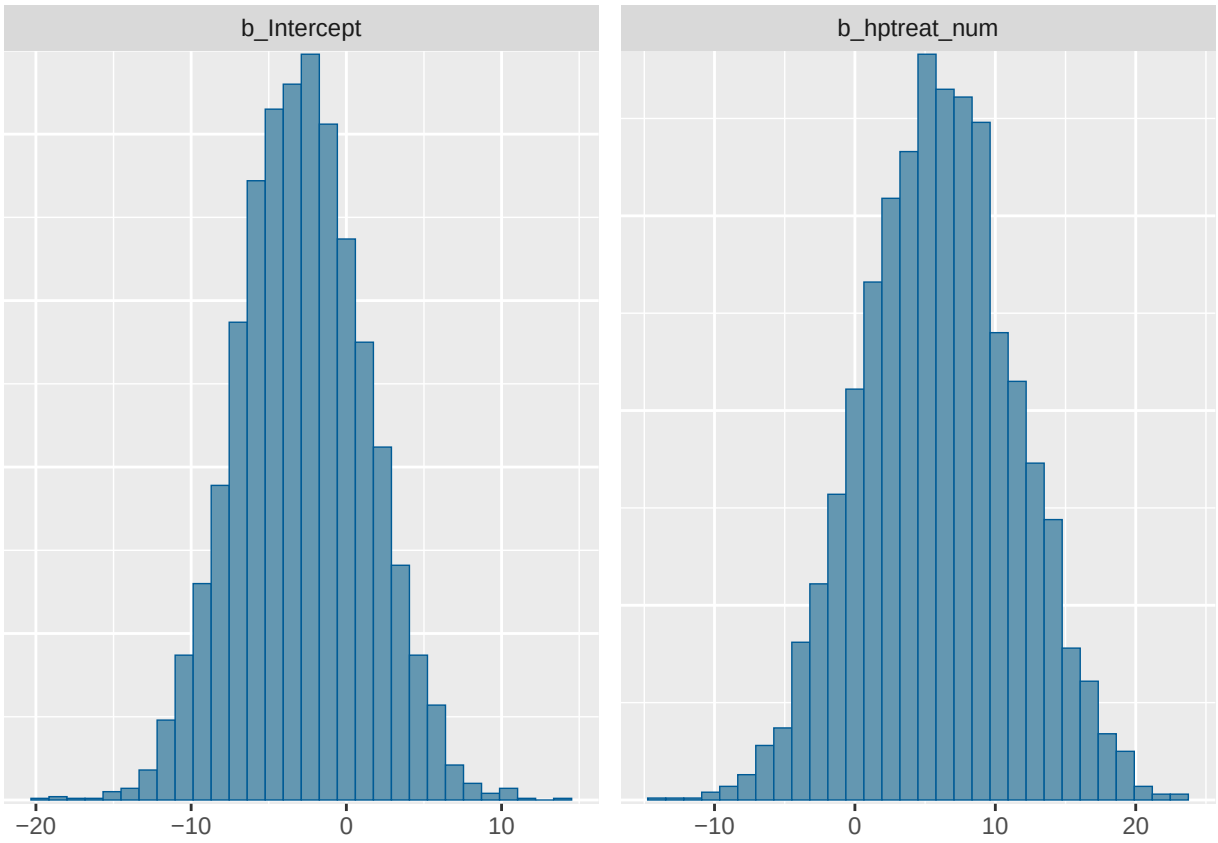

```
mcmc_trace(mfit, pars = c("b_Intercept", "b_hptreat_num"))
```




```
mcmc_hist(mfit, pars = c("b_Intercept", "b_hptreat_num"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Below from the posterior analysis we can see that the variable that encodes the change in testosterone, i.e b_hptreat_num . 95% credible interval for b_hptreat_num is almost in the positive direction indicating that The research hypothesis is true that on average, assigning a subject a high power pose vs. a low power pose will lead to higher testosterone levels after treatment.



Part 2:

2.1

```
#Exercise 2.1
# Function to calculate number of crossings
crossings_model <- function(sentence_length, alpha, beta) {
  # Calculate lambda
  lambda <- exp(alpha + beta * sentence_length)
  # Generate the number of crossings from a Poisson distribution
  rpois(1, lambda)
}
```

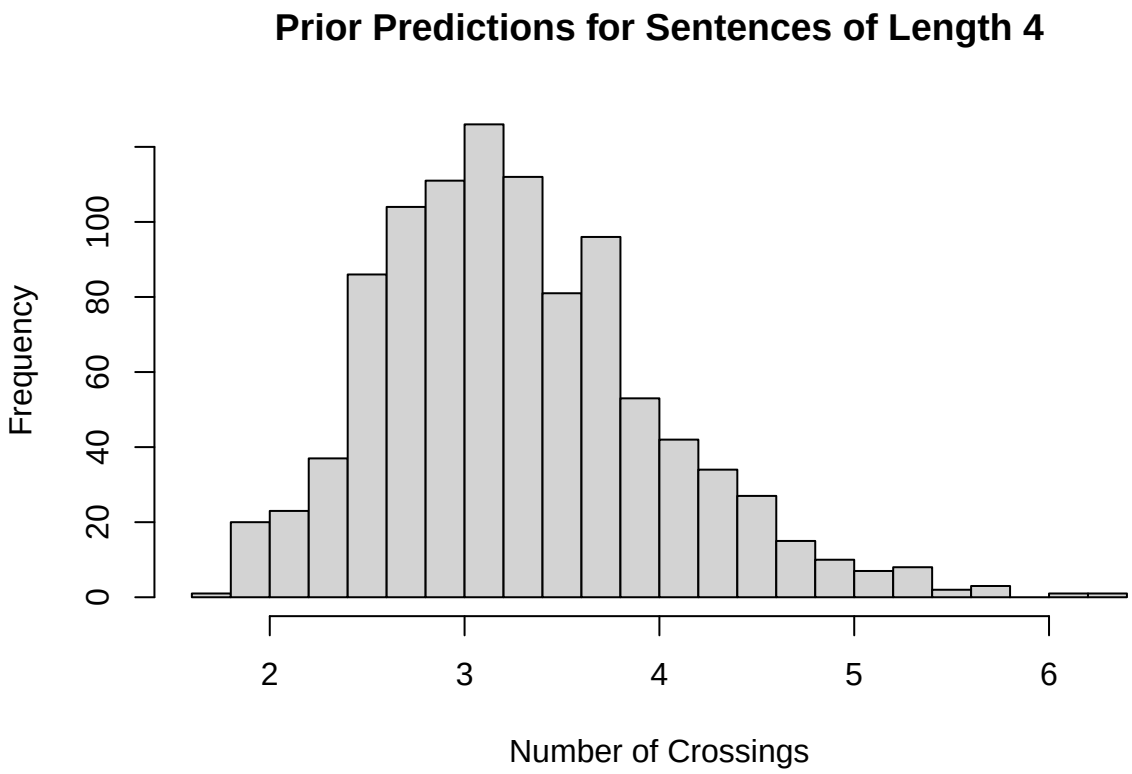
2.2

```
#Exercise 2.2
# Install and load the necessary package
if (!requireNamespace("truncnorm", quietly = TRUE)) install.packages("truncnorm")
library(truncnorm)

# Set the priors
alpha_prior_mean <- 0.15
alpha_prior_sd <- 0.1
beta_prior_mean <- 0.25
beta_prior_sd <- 0.05

# Set the lower and upper bounds for truncation
alpha_lower_bound <- 0
alpha_upper_bound <- Inf
beta_lower_bound <- 0
beta_upper_bound <- Inf
```

```r
# Number of predictions to generate
n_predictions <- 1000

# Generate prior samples for alpha and beta
set.seed(42)
alpha_samples <- rtruncnorm(n_predictions, a = alpha_lower_bound, b = alpha_upper_bound, mean = alpha_pr
beta_samples <- rtruncnorm(n_predictions, a = beta_lower_bound, b = beta_upper_bound, mean = beta_prior_

# Define the function to calculate the number of crossings
crossings_model <- function(sentence_length, alpha, beta) {
  exp(alpha + beta * sentence_length)
}

# Generate prior predictions
sentence_length <- 4
prior_predictions <- sapply(1:n_predictions, function(i) {
  crossings_model(sentence_length, alpha_samples[i], beta_samples[i])
})

# Summary of prior predictions
summary(prior_predictions)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.767   2.757   3.188   3.273   3.690   6.289
```

```r
hist(prior_predictions, breaks = 20, main = "Prior Predictions for Sentences of Length 4", xlab = "Numbe
```



2.3

```r
# Load the necessary library
library(brms)

# Load the data
data <- read.csv("crossings.csv")

# Display the first few rows of the data to understand its structure
head(data)
```

```
##   Language s.id s.length nCross
## 1   German    1        2      0
## 2   German    2        2      1
## 3   German    3        2      0
## 4   German    4        2      0
## 5   German    5        2      2
## 6   German    6        2      1
```

```r
# Define the formula for Model M1
formula_m1 <- bf(nCross ~ s.length)

# Define priors
priors_m1 <- c(
  prior(normal(0.15, 0.1), class = "Intercept"),
  prior(normal(0, 0.15), class = "b")
)
```

```r
# Fit Model M1
fit_m1 <- brm(formula_m1, data = data, family = poisson(link = "log"), prior = priors_m1, iter = 2000,
```

```
## Compiling Stan program...

## Start sampling

##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.001196 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 11.96 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 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: 1.884 seconds (Warm-up)
## Chain 1:                1.506 seconds (Sampling)
## Chain 1:                3.39 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000258 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.58 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 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: 1.863 seconds (Warm-up)
## Chain 2:                1.289 seconds (Sampling)
## Chain 2:                3.152 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.00025 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.5 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 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: 1.601 seconds (Warm-up)
## Chain 3:                1.645 seconds (Sampling)
## Chain 3:                3.246 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000251 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.51 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 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: 1.821 seconds (Warm-up)
## Chain 4:                1.544 seconds (Sampling)
## Chain 4:                3.365 seconds (Total)
```

```
## Chain 4:
```

```
# Display the summary of Model M1
summary(fit_m1)
```

```
##  Family: poisson
##   Links: mu = log
## Formula: nCross ~ s.length
##    Data: data (Number of observations: 1900)
##   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    -1.45      0.06    -1.56    -1.33 1.00     1173     1373
## s.length      0.15      0.00     0.14     0.16 1.00     1479     1829
##
## 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).
```

```
# Define the formula for Model M2
formula_m2 <- bf(nCross ~ s.length * Language)

# Define priors
priors_m2 <- c(
  prior(normal(0.15, 0.1), class = "Intercept"),
  prior(normal(0, 0.15), class = "b"),
  prior(normal(0, 0.15), class = "b", coef = "LanguageGerman"),
  prior(normal(0, 0.15), class = "b", coef = "s.length:LanguageGerman")
)

# Fit Model M2
fit_m2 <- brm(formula_m2, data = data, family = poisson(link = "log"), prior = priors_m2, iter = 2000,
```

```
## Compiling Stan program...
```

```
## Start sampling
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.00026 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.6 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
```

```
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 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: 4.899 seconds (Warm-up)
## Chain 1:                4.29 seconds (Sampling)
## Chain 1:                9.189 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000214 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.14 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 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: 14.986 seconds (Warm-up)
## Chain 2:                16.648 seconds (Sampling)
## Chain 2:                31.634 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.001056 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 10.56 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 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: 17.218 seconds (Warm-up)
## Chain 3:                17.006 seconds (Sampling)
## Chain 3:                34.224 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000832 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 8.32 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 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: 17.649 seconds (Warm-up)
## Chain 4:                14.187 seconds (Sampling)
## Chain 4:                31.836 seconds (Total)
## Chain 4:
```

```
# Display the summary of Model M2
summary(fit_m2)
```

```
##  Family: poisson
##   Links: mu = log
## Formula: nCross ~ s.length * Language
##    Data: data (Number of observations: 1900)
##   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
## Intercept                  -1.10      0.08    -1.25    -0.95 1.01     1460
## s.length                    0.11      0.01     0.10     0.12 1.00     1514
## LanguageGerman             -0.62      0.09    -0.81    -0.43 1.00     1415
## s.length:LanguageGerman     0.07      0.01     0.06     0.08 1.00     1356
##                         Tail_ESS
## Intercept                   1926
## s.length                    1932
## LanguageGerman              1676
```

```
## s.length:LanguageGerman     1542
##
## 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).
```

```
# Load the bayesplot package for plotting
library(bayesplot)

# Convert brmsfit objects to mcmc objects for plotting
mfit_m1 <- as.mcmc(fit_m1)
```

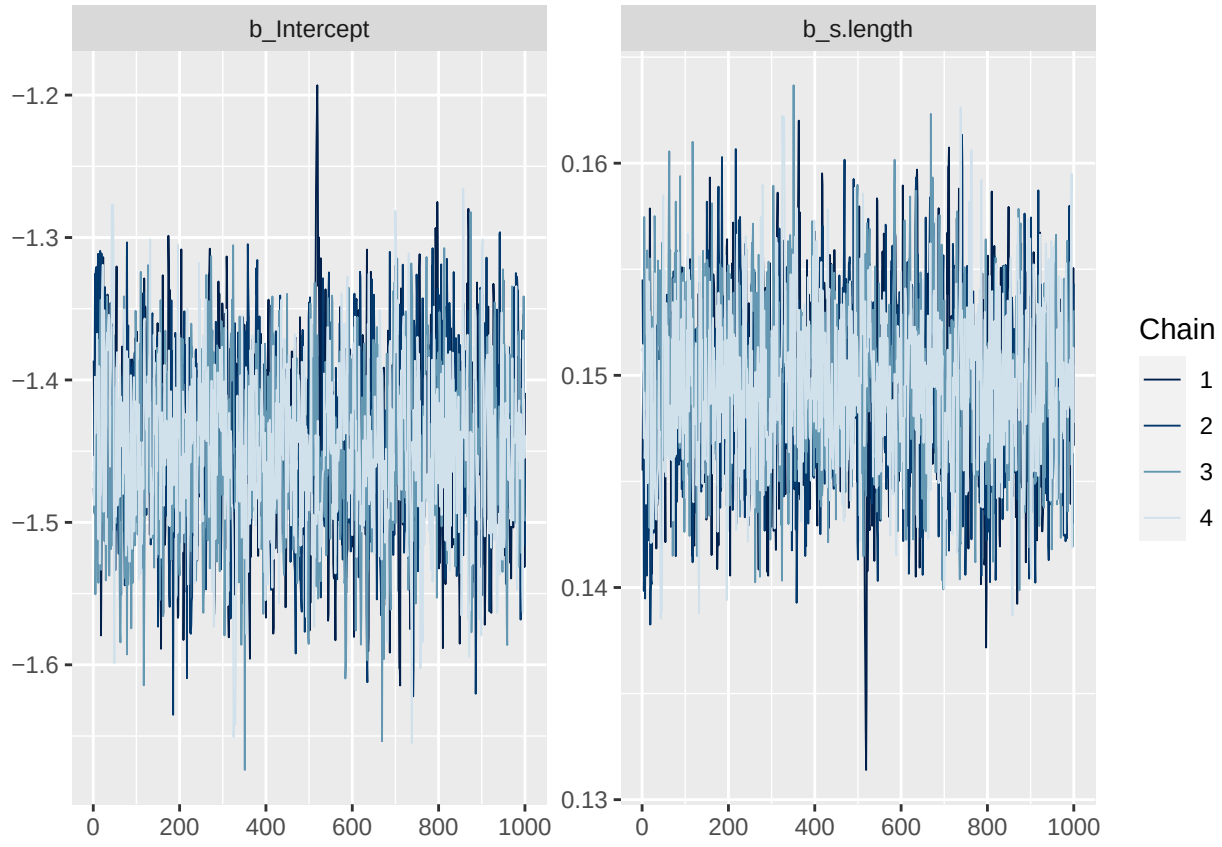
```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

```
mfit_m2 <- as.mcmc(fit_m2)
```

```
## Warning: as.mcmc.brmsfit is deprecated and will eventually be removed.
```

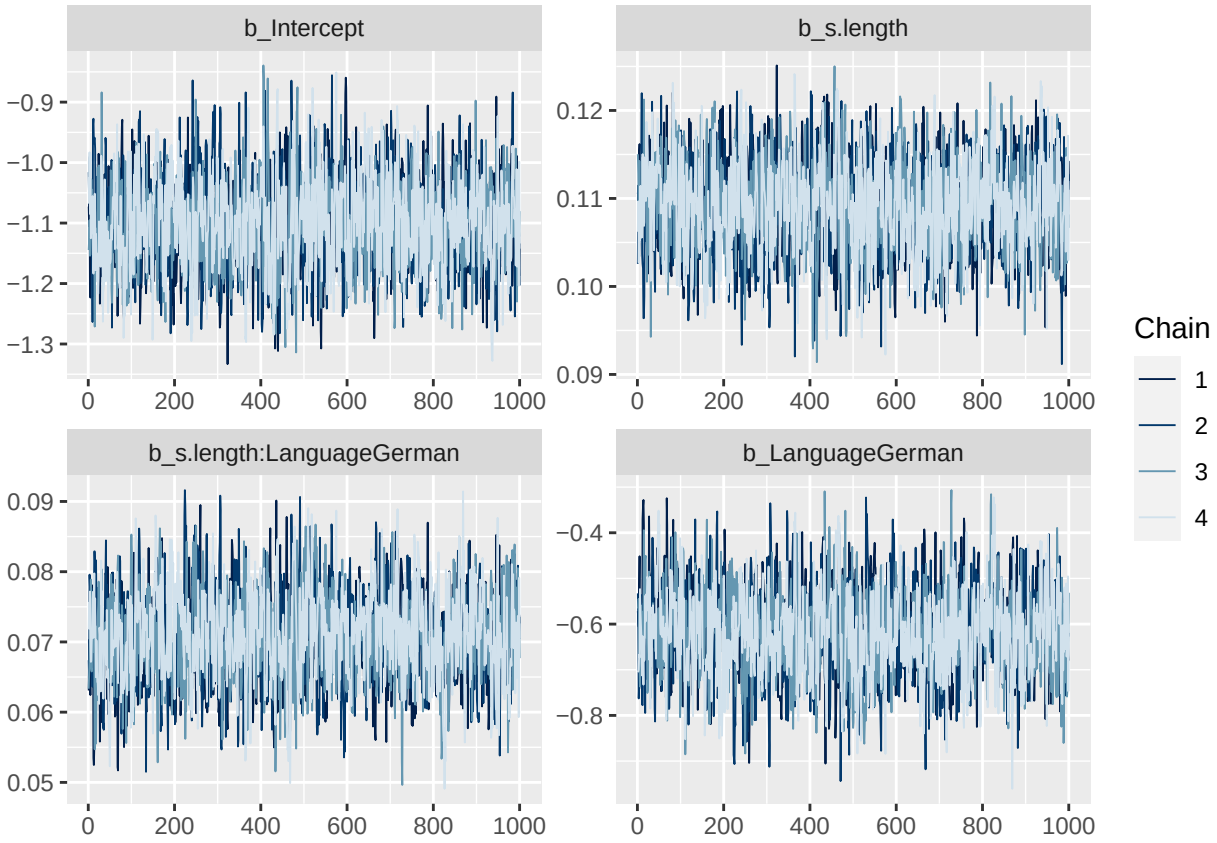
```
# Plot trace plots for Model M1
mcmc_trace(mfit_m1, pars = c("b_Intercept", "b_s.length"))
```

2.3




```
# Plot trace plots for Model M2
mcmc_trace(mfit_m2, pars = c("b_Intercept", "b_s.length", "b_s.length:LanguageGerman", "b_LanguageGerma
```

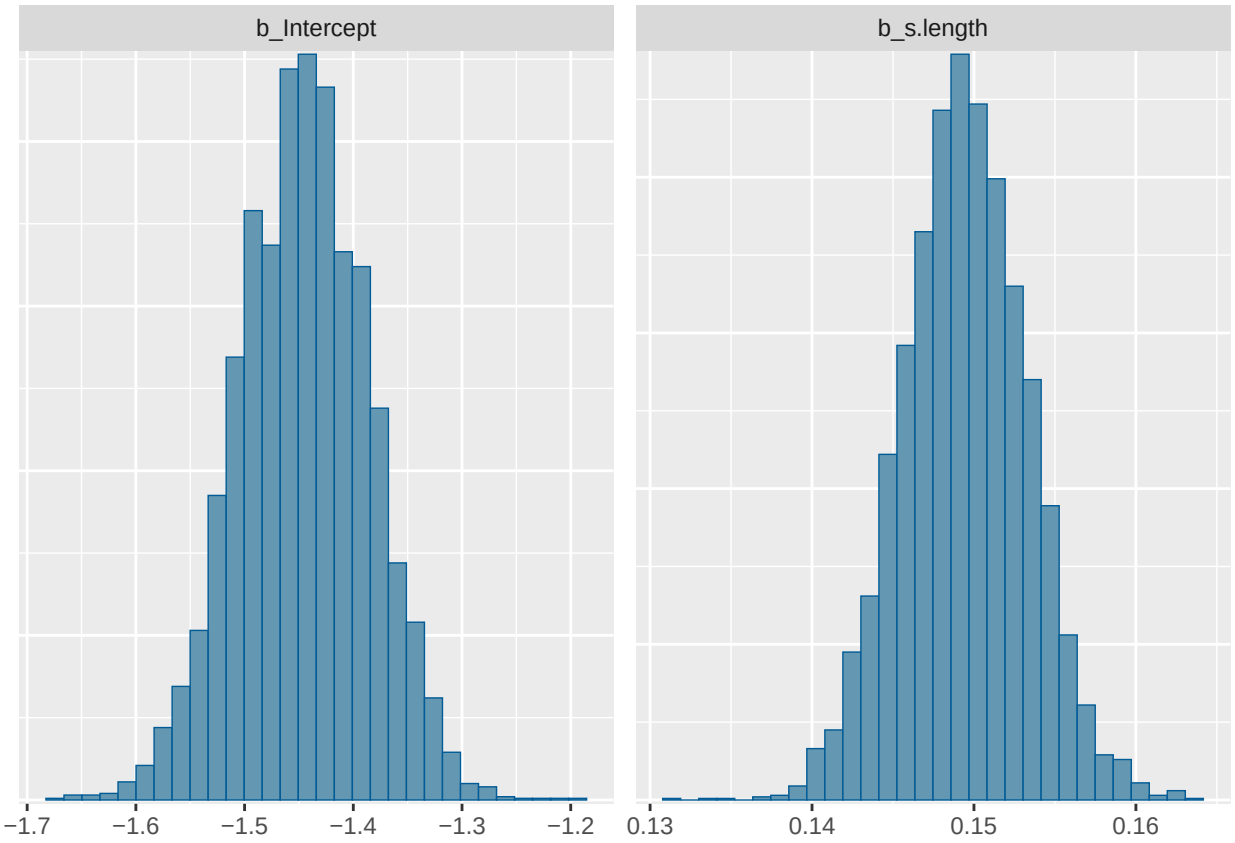
2.3




```
# Plot histograms for Model M1
mcmc_hist(mfit_m1, pars = c("b_Intercept", "b_s.length"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

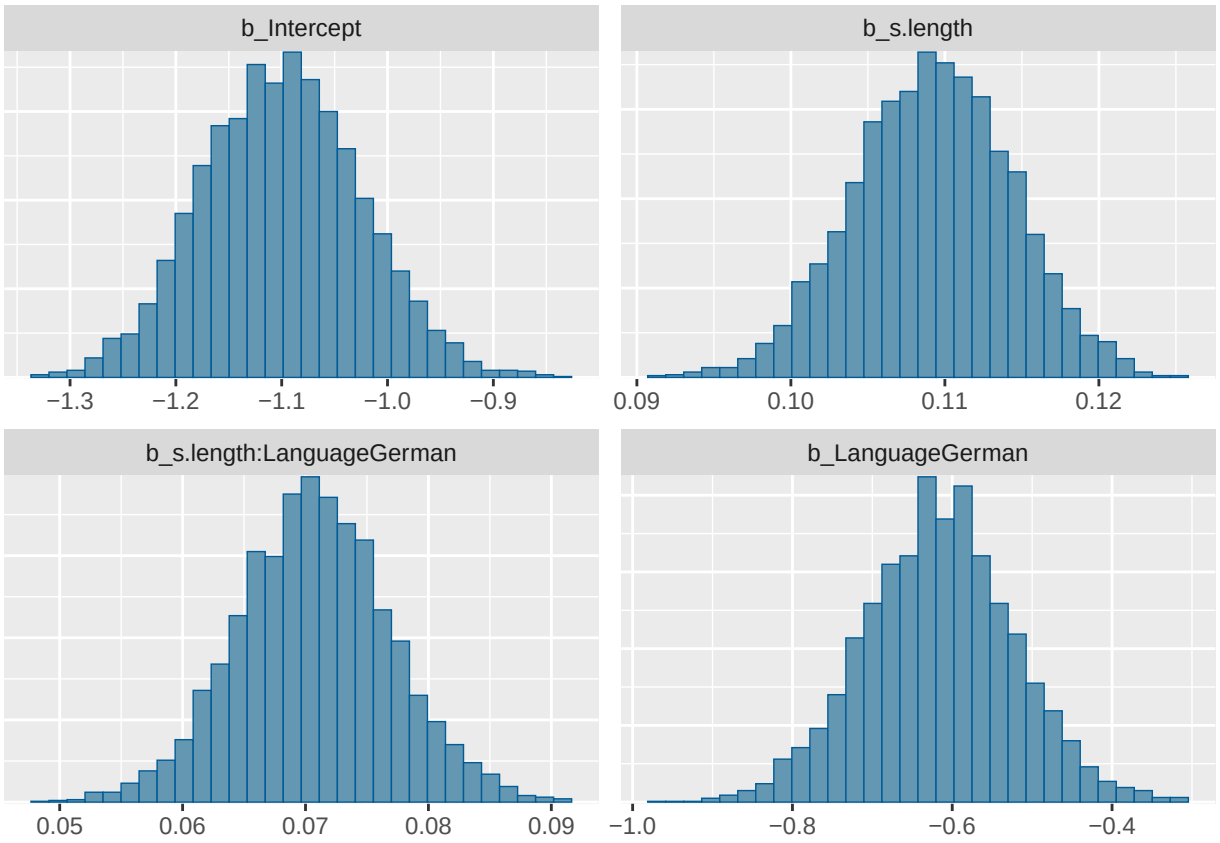
2.3




```
# Plot histograms for Model M2
mcmc_hist(mfit_m2, pars = c("b_Intercept", "b_s.length", "b_s.length:LanguageGerman", "b_LanguageGerman
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

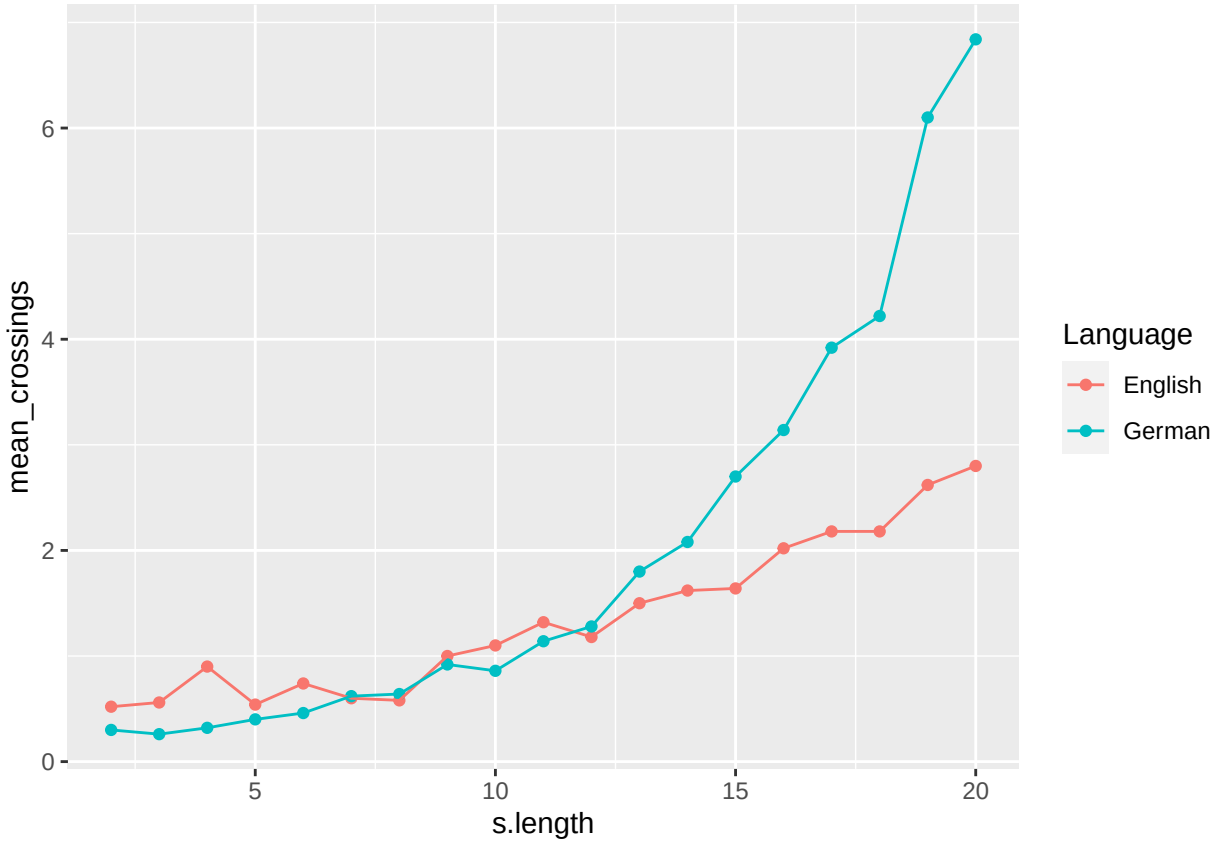
2.3



2.4

```
#Exercise 2.4
# Load the necessary libraries
library(brms)
library(dplyr)
library(ggplot2)
# Load the data
observed <- read.table("crossings.csv", sep=",", header=TRUE)

# Visualize average rate of crossings
observed %>% group_by(Language, s.length) %>%
  summarise(mean_crossings = mean(nCross)) %>%
  ggplot(aes(x = s.length, y = mean_crossings, group = Language, color = Language)) +
  geom_point() + geom_line()
```

```
## `summarise()` has grouped output by 'Language'. You can override using the
## `.groups` argument.
```




```
# Center the sentence length predictor
observed$s.length <- observed$s.length - mean(observed$s.length)

# Create an indicator variable for language (1 for German, 0 for English)
observed$lang <- ifelse(observed$Language == "German", 1, 0)
```

```
# Vectors to store log predictive densities for each fold
lpds.m1 <- c()
lpds.m2 <- c()

# Create a copy of the dataset to keep track of untested data
untested <- observed

# Number of folds
k_folds <- 5

# Perform k-fold cross-validation
for(k in 1:k_folds) {
  # Prepare test data and training data
  ytest <- sample_n(untested, size = nrow(observed) / k_folds)
  ytrain <- setdiff(observed, ytest)
  untested <- setdiff(untested, ytest)

  # Fit Model M1 on training data
  fit.m1 <- brm(nCross ~ 1 + s.length, data = ytrain,
                family = poisson(link = "log"),
```

```
                prior = c(prior(normal(0.15, 0.1), class = "Intercept"),
                          prior(normal(0, 0.15), class = "b")),
                cores = 4)

  # Fit Model M2 on training data
  fit.m2 <- brm(nCross ~ 1 + s.length + lang + s.length * lang, data = ytrain,
                family = poisson(link = "log"),
                prior = c(prior(normal(0.15, 0.1), class = "Intercept"),
                          prior(normal(0, 0.15), class = "b")),
                cores = 4)

  # Retrieve posterior samples
  post.m1 <- posterior_samples(fit.m1)
  post.m2 <- posterior_samples(fit.m2)

  # Calculate log pointwise predictive density using test data
  lppd.m1 <- 0
  lppd.m2 <- 0

  for(i in 1:nrow(ytest)) {
    lpd_im1 <- log(mean(dpois(ytest[i,]$nCross,
                              lambda = exp(post.m1[,1] +
                                           post.m1[,2] * ytest[i,]$s.length))))
    lppd.m1 <- lppd.m1 + lpd_im1

    lpd_im2 <- log(mean(dpois(ytest[i,]$nCross,
                              lambda = exp(post.m2[,1] +
                                           post.m2[,2] * ytest[i,]$s.length +
                                           post.m2[,3] * ytest[i,]$lang +
                                           post.m2[,4] * ytest[i,]$s.length * ytest[i,]$lang))))
    lppd.m2 <- lppd.m2 + lpd_im2
  }

  # Store log predictive densities
  lpds.m1 <- c(lpds.m1, lppd.m1)
  lpds.m2 <- c(lpds.m2, lppd.m2)
}
```

```
## Compiling Stan program...

## Start sampling

## Compiling Stan program...

## Start sampling

## Warning: Method 'posterior_samples' is deprecated. Please see ?as_draws for
## recommended alternatives.

## Warning: Method 'posterior_samples' is deprecated. Please see ?as_draws for
## recommended alternatives.
```

```
## Compiling Stan program...
## Start sampling

## Compiling Stan program...

## Start sampling

## Warning: Method 'posterior_samples' is deprecated. Please see ?as_draws for
## recommended alternatives.

## Warning: Method 'posterior_samples' is deprecated. Please see ?as_draws for
## recommended alternatives.

## Compiling Stan program...
## Start sampling

## Compiling Stan program...

## Start sampling

## Warning: Method 'posterior_samples' is deprecated. Please see ?as_draws for
## recommended alternatives.

## Warning: Method 'posterior_samples' is deprecated. Please see ?as_draws for
## recommended alternatives.

## Compiling Stan program...
## Start sampling

## Compiling Stan program...

## Start sampling

## Warning: Method 'posterior_samples' is deprecated. Please see ?as_draws for
## recommended alternatives.

## Warning: Method 'posterior_samples' is deprecated. Please see ?as_draws for
## recommended alternatives.

## Compiling Stan program...
## Start sampling

## Compiling Stan program...

## Start sampling

## Warning: Method 'posterior_samples' is deprecated. Please see ?as_draws for
## recommended alternatives.

## Warning: Method 'posterior_samples' is deprecated. Please see ?as_draws for
## recommended alternatives.
```

2.4

```r
# Calculate the expected log predictive density (elpd) for each model
elpd.m1 <- sum(lpds.m1)
elpd.m2 <- sum(lpds.m2)

# Print predictive accuracy of both models
cat("Predictive accuracy of model M1: ", elpd.m1, "\n")
```

```
## Predictive accuracy of model M1:  -2814.542
```

```r
cat("Predictive accuracy of model M2: ", elpd.m2, "\n")
```

```
## Predictive accuracy of model M2:  -2680.397
```

```r
# Evidence in favor of M2 over M1
difference_elpd <- elpd.m2 - elpd.m1
cat("Evidence in favor of M2 over M1 (difference in elpd): ", difference_elpd, "\n")
```

```
## Evidence in favor of M2 over M1 (difference in elpd):  134.1451
```