CGS698
Assignment-4

Anupam Chaudhary
(210170)

Part 1:A simple linear regression: Power posing and testosterone

```
> library(readr)
> library(dplyr)
> library(brms)
> library(rstudioapi)
> # Load the observed data
> df_powerpose <- read.csv("F:/Study
Material/Summer/CGS698/Assignment/df_powerpose.csv", sep = ",", header = TRUE)
> df_powerpose$hptreat <- as.factor(df_powerpose$hptreat)
> # Convert the gender variable to numeric (0 = Female, 1 = Male)
> df_powerpose$female <- ifelse(df_powerpose$female == "Female", 1, 0)
> # Define the formula for the linear model
> formula <- testm2 ~ hptreat + female + age
> # Define weakly informative priors
> priors <- c(
+  set_prior("normal(0, 1)", class = "b"),   # Priors for the regression
coefficients
+  set_prior("normal(0, 1)", class = "Intercept")  # Prior for the intercept
+ )
> model <- brm(
+  formula = formula,
+  data = df_powerpose,
+  family = gaussian(),
+  prior = priors,
+  iter = 2000,
+  chains = 4,
+  cores = 4
+ )
Compiling Stan program...
Start sampling
> summary(model)
```

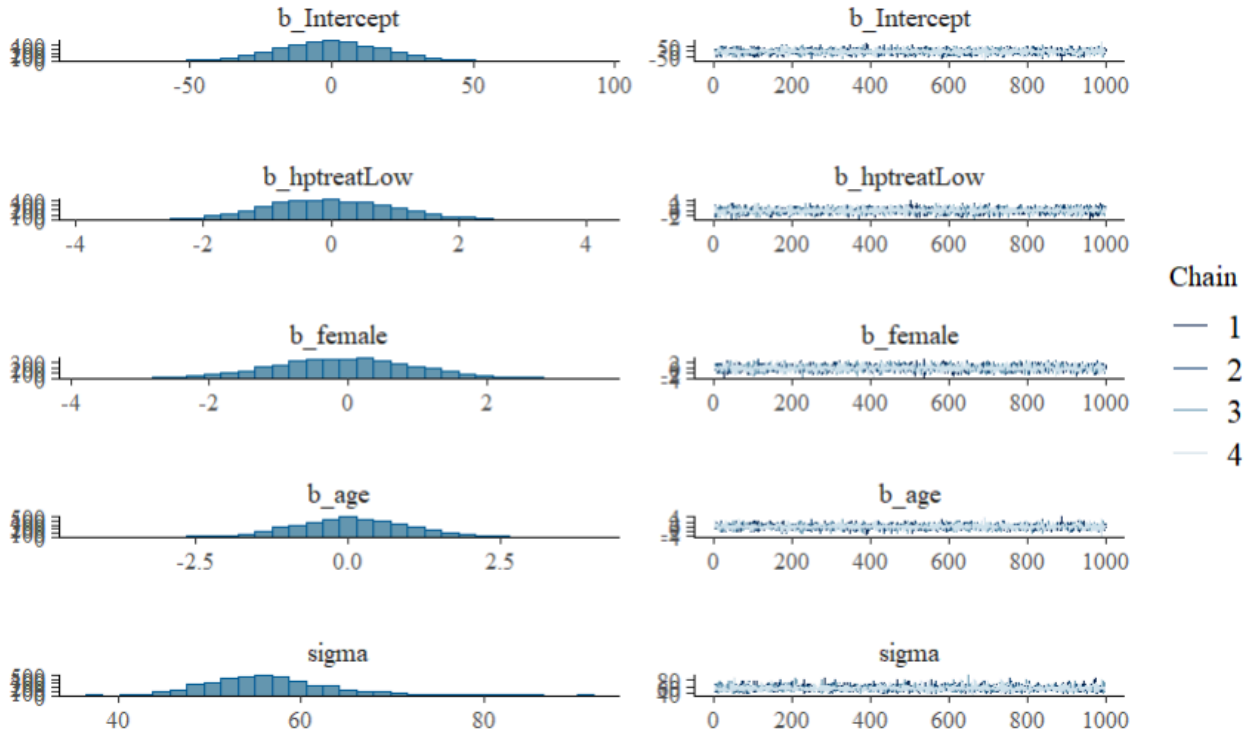
Summary of Model:

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: testm2 ~ hptreat + female + age
   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     -0.14     19.87   -39.87    39.09 1.00     5256     2830
hptreatLow    -0.03      1.00    -2.02     1.93 1.00     4707     3065
female        -0.06      1.01    -2.02     1.90 1.00     4960     3132
age            0.04      0.93    -1.78     1.88 1.00     5277     2949

Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma    56.45      6.62    45.42    71.58 1.00     5040     3010

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:

```
> plot(model)
```



Part 2:Poisson regression models and hypothesis testing

2.1)

```
> library(MASS)
> # Function to predict the number of crossing dependencies
> predict_crossings <- function(sentence_length, alpha, beta) {
+  # Compute the log of the expected rate
+  log_lambda <- alpha + beta * sentence_length
+
+  # Convert to the expected rate
+  lambda <- exp(log_lambda)
+
+  # Generate the number of crossing dependencies from the Poisson distribution
+  crossings <- rpois(1, lambda)
+
+  return(crossings)
+ }
> # Example usage
> set.seed(123)  # Set seed for reproducibility
> sentence_length <- 15
> alpha <- 1
> beta <- 0.1
> predicted_crossings <- predict_crossings(sentence_length, alpha, beta)
> cat("Predicted number of crossing dependencies:", predicted_crossings, "\n")
```

Output:

```
Predicted number of crossing dependencies: 10
```

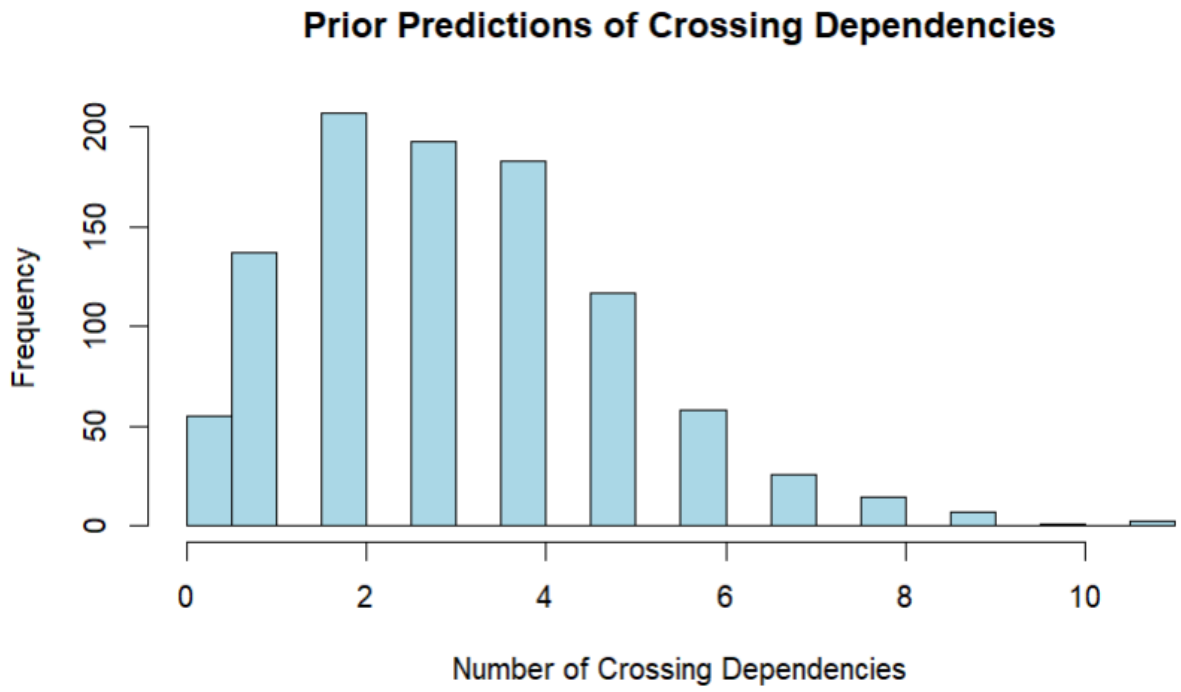
2.2)

```
> library(MASS)
> # Function to generate prior predictions
> generate_prior_predictions <- function(sentence_length, n_samples = 1000) {
+  # Sample from the priors
+  alpha_samples <- rnorm(n_samples, mean = 0.15, sd = 0.1)
+  beta_samples <- rnorm(n_samples, mean = 0.25, sd = 0.05)
+
+  # Compute the log of the expected rate for each sample
+  log_lambda_samples <- alpha_samples + beta_samples * sentence_length
+
+  # Convert to the expected rate
+  lambda_samples <- exp(log_lambda_samples)
+
+  # Generate the number of crossing dependencies for each sample
+  crossing_samples <- rpois(n_samples, lambda_samples)
+
+  return(crossing_samples)
+ }
> # Generate prior predictions for sentence length of 4
> set.seed(123)  # Set seed for reproducibility
```

```
> sentence_length <- 4
> n_samples <- 1000
> prior_predictions <- generate_prior_predictions(sentence_length, n_samples)
> # Summarize and plot the prior predictions
> summary(prior_predictions)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   2.000   3.000   3.184   4.000  11.000
> hist(prior_predictions, breaks = 30, main = "Prior Predictions of Crossing
Dependencies",
+     xlab = "Number of Crossing Dependencies", col = "lightblue", border =
"black")
```

Result:

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.000   2.000   3.000   3.184   4.000  11.000
```



2.3)

```
> library(brms)
> # Set the working directory
> setwd("F:\\Study Material\\Summer\\CGS698\\Assignment")
> # Load the data
> crossings_data <- read.csv("crossings.csv")
> # Inspect the data
> head(crossings_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
> str(crossings_data)
'data.frame':     1900 obs. of  4 variables:
 $ Language: chr  "German" "German" "German" "German" ...
 $ s.id    : int  1 2 3 4 5 6 7 8 9 10 ...
 $ s.length: int  2 2 2 2 2 2 2 2 2 2 ...
 $ nCross  : int  0 1 0 0 2 1 1 0 0 0 ...
> summary(crossings_data)
   Language             s.id          s.length      nCross
 Length:1900        Min.   :  1.0   Min.   : 2   Min.   : 0.000
 Class :character   1st Qu.:238.0   1st Qu.: 6   1st Qu.: 0.000
 Mode  :character   Median :475.5   Median :11   Median : 1.000
                    Mean   :475.5   Mean   :11   Mean   : 1.674
                    3rd Qu.:713.0   3rd Qu.:16   3rd Qu.: 2.000
                    Max.   :950.0   Max.   :20   Max.   :12.000
> # Define Model M1 formula
> formula_M1 <- bf(nCross ~ s.length + (1 | Language), family = poisson())
> # Prior for Model M1
> priors_M1 <- c(
+  prior(normal(0.15, 0.1), class = "Intercept"),  # alpha
+  prior(normal(0, 0.15), class = "b")             # beta
+ )
> # Define Model M2 formula
> formula_M2 <- bf(nCross ~ s.length * Language + (1 | Language), family =
poisson())
> # Prior for Model M2
> priors_M2 <- c(
+  prior(normal(0.15, 0.1), class = "Intercept"),   # alpha
+  prior(normal(0, 0.15), class = "b"),             # beta
+  prior(normal(0, 0.15), class = "b", coef = "LanguageGerman"),  # beta for
languageGerman
+  prior(normal(0, 0.15), class = "b", coef = "s.length:LanguageGerman")  #
beta for interaction term
+ )
> # Fit Model M1
```

```
> fit_M1 <- brm(formula_M1, data = crossings_data, prior = priors_M1, iter =
2000, chains = 4)
Compiling Stan program...
Start sampling

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000442 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take
4.42 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) .... (and so on)

> # Fit Model M2
> fit_M2 <- brm(formula_M2, data = crossings_data, prior = priors_M2, iter =
2000, chains = 4)
Compiling Stan program...
Start sampling

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000403 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take
4.03 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)...(and so on)


> # Summarize Model M1
> summary(fit_M1)
> # Summarize Model M2
> summary(fit_M2)
```

Summary of Model M1:

```
Family: poisson
  Links: mu = log
Formula: nCross ~ s.length + (1 | Language)
   Data: crossings_data (Number of observations: 1900)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000
```

```
Multilevel Hyperparameters:
~Language (Number of levels: 2)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.60      0.48     0.13     1.69 1.32       10       17

Regression Coefficients:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept    -1.49      0.09    -1.68    -1.31 1.05      814     1354
s.length      0.15      0.00     0.14     0.16 1.09       29     1353

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).
```

Summary of Model M2:

```
 Family: poisson
  Links: mu = log
Formula: nCross ~ s.length * Language + (1 | Language)
   Data: crossings_data (Number of observations: 1900)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Multilevel Hyperparameters:
~Language (Number of levels: 2)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.94      0.56     0.27     2.41 1.00      710      584

Regression Coefficients:
                      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
Tail_ESS
Intercept                -1.43      0.13    -1.69    -1.17 1.00     2380
2012
s.length                  0.10      0.01     0.09     0.11 1.00     2256
2212
LanguageGerman           -0.03      0.15    -0.34     0.27 1.00     1934
2234
s.length:LanguageGerman   0.09      0.01     0.08     0.11 1.00     2038
2053
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).
```

2.4)

```
> library(brms)
> library(dplyr)
> library(ggplot2)
> # Load the dataset
> observed <- read.csv("F:/Study
Material/Summer/CGS698/Assignment/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 predictors
> observed$s.length <- observed$s.length - mean(observed$s.length)
>
> # Create indicator variable for language
> observed$lang <- ifelse(observed$Language == "German", 1, 0)
> # Initialize vectors to store log predictive densities in each fold
> lpds.m1 <- c()
> lpds.m2 <- c()
>
> # Initialize untested data
> untested <- observed
>
> # Perform k-fold cross-validation (assuming k = 4)
> for (k in 1:4) {
+  # Prepare test data and training data
+  ytest <- sample_n(untested, size = nrow(observed) / 5)
+  ytrain <- setdiff(observed, ytest)
+  untested <- setdiff(untested, ytest)
+
+  # Fit model M1 on training data
+  fit.m1 <- brm(
+    nCross ~ 1 + s.length,
+    data = ytrain,
```

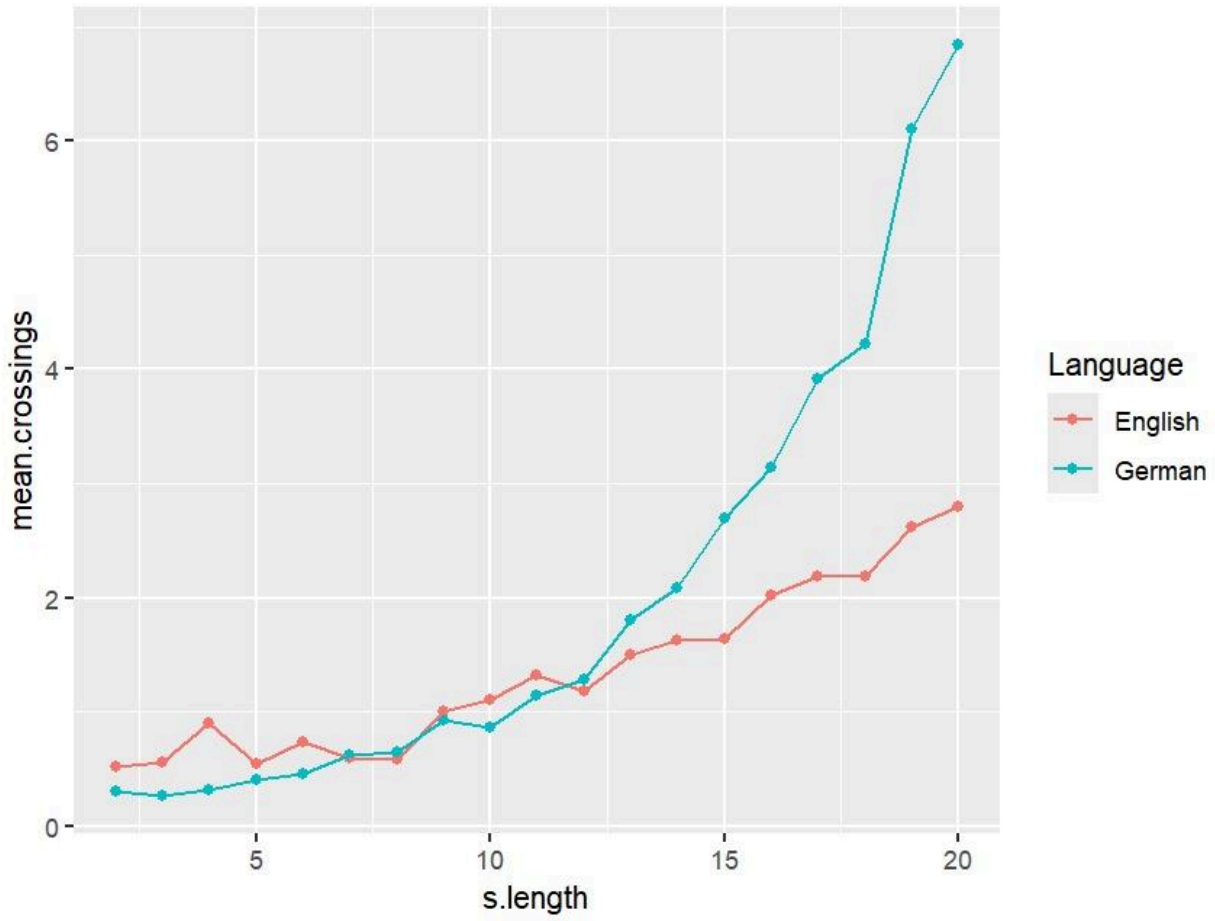
```
+    family = poisson(link = "log"),
+    prior = c(
+      prior(normal(0.15, 0.1), class = Intercept),
+      prior(normal(0, 0.15), class = b)
+    ),
+    cores = 4
+  )
+
+  # Fit model M2 on training data
+  fit.m2 <- brm(
+    nCross ~ 1 + s.length + lang + s.length * lang,
+    data = ytrain,
+    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 for model M1
+  lppd.m1 <- 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
+  }
+
+  # Calculate log pointwise predictive density using test data for model M2
+  lppd.m2 <- 0
+  for (i in 1:nrow(ytest)) {
+    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
Compiling Stan program...
Start sampling
Compiling Stan program...
Start sampling
Compiling Stan program...
Start sampling
Compiling Stan program...
Start sampling
Compiling Stan program...
Start sampling
Compiling Stan program...
Start sampling
Warning messages:
1: Method 'posterior_samples' is deprecated. Please see ?as_draws for
recommended alternatives.
2: Method 'posterior_samples' is deprecated. Please see ?as_draws for
recommended alternatives.
3: Method 'posterior_samples' is deprecated. Please see ?as_draws for
recommended alternatives.
4: Method 'posterior_samples' is deprecated. Please see ?as_draws for
recommended alternatives.
5: Method 'posterior_samples' is deprecated. Please see ?as_draws for
recommended alternatives.
6: Method 'posterior_samples' is deprecated. Please see ?as_draws for
recommended alternatives.
7: Method 'posterior_samples' is deprecated. Please see ?as_draws for
recommended alternatives.
8: Method 'posterior_samples' is deprecated. Please see ?as_draws for
recommended alternatives.
>
> # Calculate expected log predictive density (elpd) for model M1 and M2
> elpd.m1 <- sum(lpds.m1)
> elpd.m2 <- sum(lpds.m2)
>
> # Calculate evidence in favor of M2 over M1
> difference_elpd <- elpd.m2 - elpd.m1
>
> # Print results
```

Output:

```
> cat("Predictive accuracy of model M1 (elpd):", elpd.m1, "\n")
Predictive accuracy of model M1 (elpd): -2224
> cat("Predictive accuracy of model M2 (elpd):", elpd.m2, "\n")
Predictive accuracy of model M2 (elpd): -2131.142
> cat("Difference in elpd (M2 - M1):", difference_elpd, "\n")
Difference in elpd (M2 - M1): 92.85859
```

Plot:



>