Assignment 4

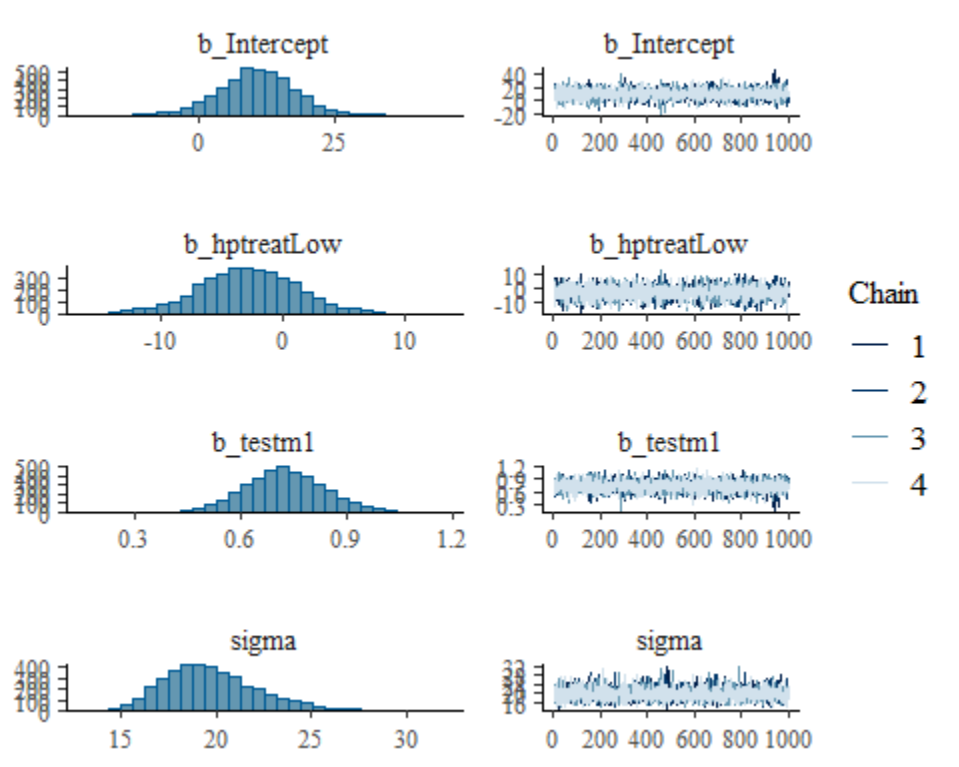
Aryan Thada : 210205

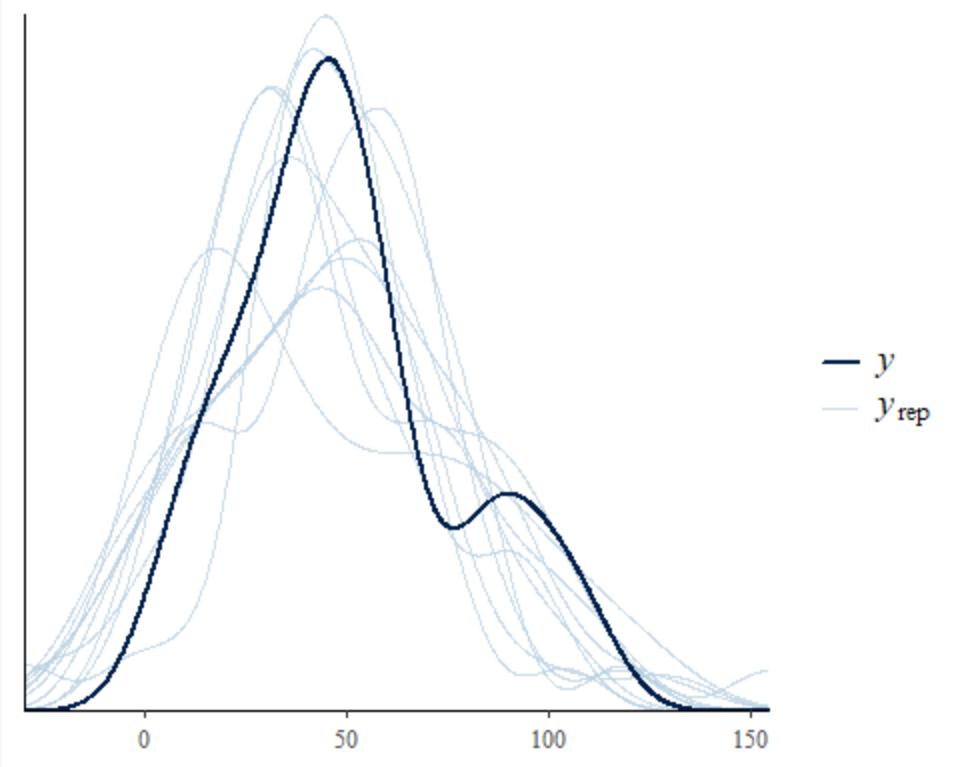
2024-07-03

{Q1}

```
#install.packages("readr")
library(readr)
library(dplyr)
library(brms)



df_powerpose <- read_csv("df_powerpose.csv")



head(df_powerpose)




priors <- c(
  set_prior("normal(0, 10)", class = "Intercept"),  # Prior for the intercept
  set_prior("normal(0, 5)", class = "b"),          # Prior for the slopes
  set_prior("cauchy(0, 2)", class = "sigma")       # Prior for the residual standard deviation
)
formula <- bf(testm2 ~ hptreat + testm1)
fit <- brm(formula, data = df_powerpose, prior = priors,
           chains = 4, iter = 2000, warmup = 1000, seed = 123)
```

```
summary(fit)

plot(fit)

pp_check(fit)


summary(fit)
```






```
Family: gaussian
  Links: mu = identity; sigma = identity
Formula: testm2 ~ hptreat + testm1
   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
Intercept     10.60      6.97    -3.50    23.88 1.00
hptreatLow    -2.80      4.05   -11.18     5.38 1.00
testm1         0.72      0.12     0.49     0.96 1.00
           Bulk_ESS Tail_ESS
Intercept      4168     3058
hptreatLow     4042     2874
testm1         4359     3034

Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
sigma    19.79      2.54    15.61    25.37 1.00     3266
      Tail_ESS
sigma     3068

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).
```

{Q2}

{r}

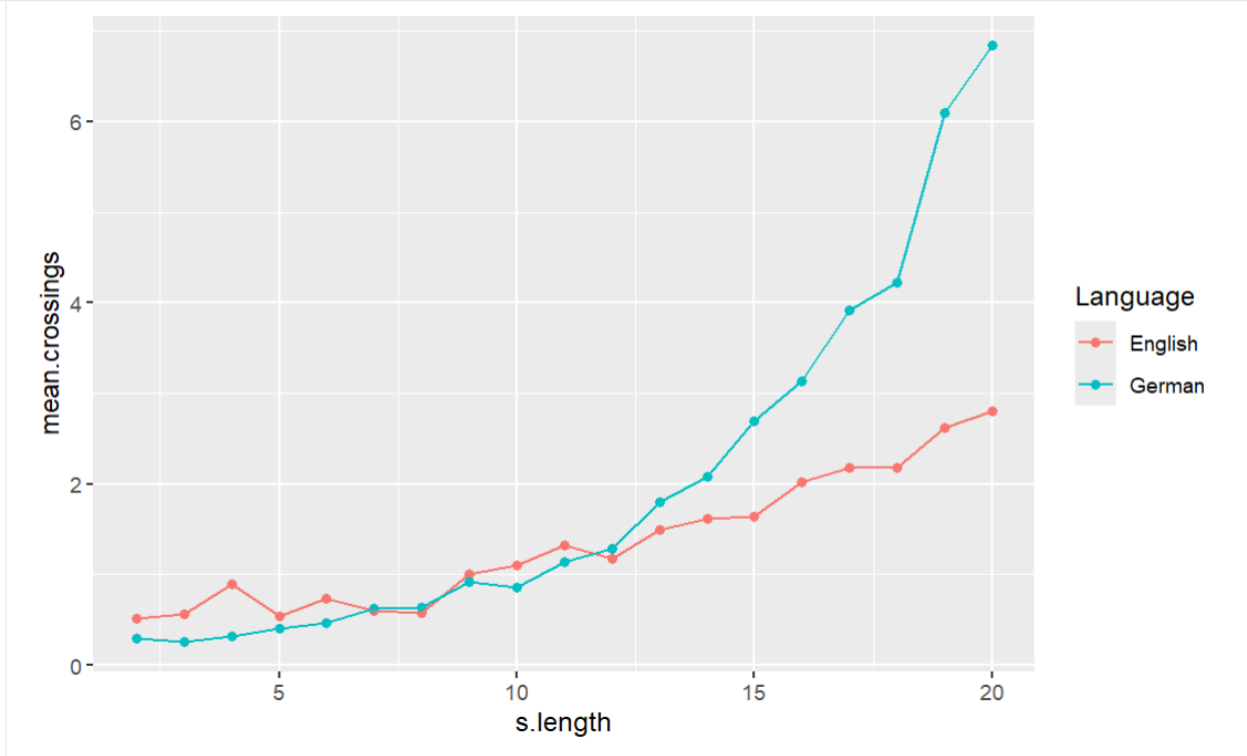
```
number_of_crossings <- function(sentence_length, alpha, beta)
 {

 log_lambda_i <- alpha + beta * sentence_length
 lambda_i <- exp(log_lambda_i)



 number_of_crossings <- rpois(1, lambda_i)

 return(number_of_crossings)
}



generate_prior_predictions <- function(n_samples,
sentence_length,alpha_samples,beta_samples)
 {




 crossings <- numeric(n_samples)


   for (i in 1:n_samples)
```

```
  {
    alpha <- alpha_samples[i]
    beta <- beta_samples[i]

    log_lambda_i <- alpha + beta * sentence_length
    lambda_i <- exp(log_lambda_i)

    crossings[i] <- rpois(1, lambda_i)
  }

 return(crossings)
}

n_samples <- 1000
sentence_length <- 4
alpha_samples <- rnorm(n_samples, mean = 0.15, sd = 0.1)
beta_samples <- rnorm(n_samples, mean = 0.25, sd = 0.05)



#crossings <- number_of_crossings(sentence_length, alpha, beta)
prior_predictions <- generate_prior_predictions(n_samples,
sentence_length,alpha_samples,beta_samples)



summary(prior_predictions)
```

```r
library(brms)
library(dplyr)
library(ggplot2)

data <- read.csv("crossings.csv")

str(data)

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

formula_M1 <- bf(nCross ~ s.length)

fit_M1 <- brm(formula_M1,
              data = data,
              family = poisson(),
              prior = priors,
              chains = 4,
              iter = 2000,
              warmup = 1000)

formula_M2 <- bf(nCross ~ s.length + Language + s.length:Language)
```

```
fit_M2 <- brm(formula_M2,
              data = data,
              family = poisson(),
              prior = priors,
              chains = 4,
              iter = 2000,
              warmup = 1000)

summary(fit_M1)
summary(fit_M2)




library(brms)
library(dplyr)
library(ggplot2)

observed <- read.csv("crossings.csv")

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()

observed$s.length <- observed$s.length - mean(observed$s.length)
```

```
observed$lang <- ifelse(observed$Language == "German", 1, 0)
```




```
lpds.m1 <- c()
lpds.m2 <- c()
untested <- observed


for (k in 1:5)
 {

  ytest <- sample_n(untested, size = nrow(observed) / 5)
  ytrain <- setdiff(observed, ytest)
  untested <- setdiff(untested, ytest)
```

```
 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.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)


  post.m1 <- posterior_samples(fit.m1)
  post.m2 <- posterior_samples(fit.m2)


  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
 }
 lpds.m1 <- c(lpds.m1, lppd.m1)
 lpds.m2 <- c(lpds.m2, lppd.m2)
}


# Predictive accuracy of model M1
elpd.m1 <- sum(lpds.m1)
# Predictive accuracy of model M2
elpd.m2 <- sum(lpds.m2)
# Evidence in favor of M2 over M1
difference_elpd <- elpd.m2 - elpd.m1

print(paste("Predictive accuracy of model M1 (elpd.m1):", elpd.m1))
print(paste("Predictive accuracy of model M2 (elpd.m2):", elpd.m2))
print(paste("Evidence in favor of M2 over M1 (difference_elpd):", difference_elpd))
```