Assignment – 4

Name:- Pritam Priyadarshi
Roll No.:- 230793

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

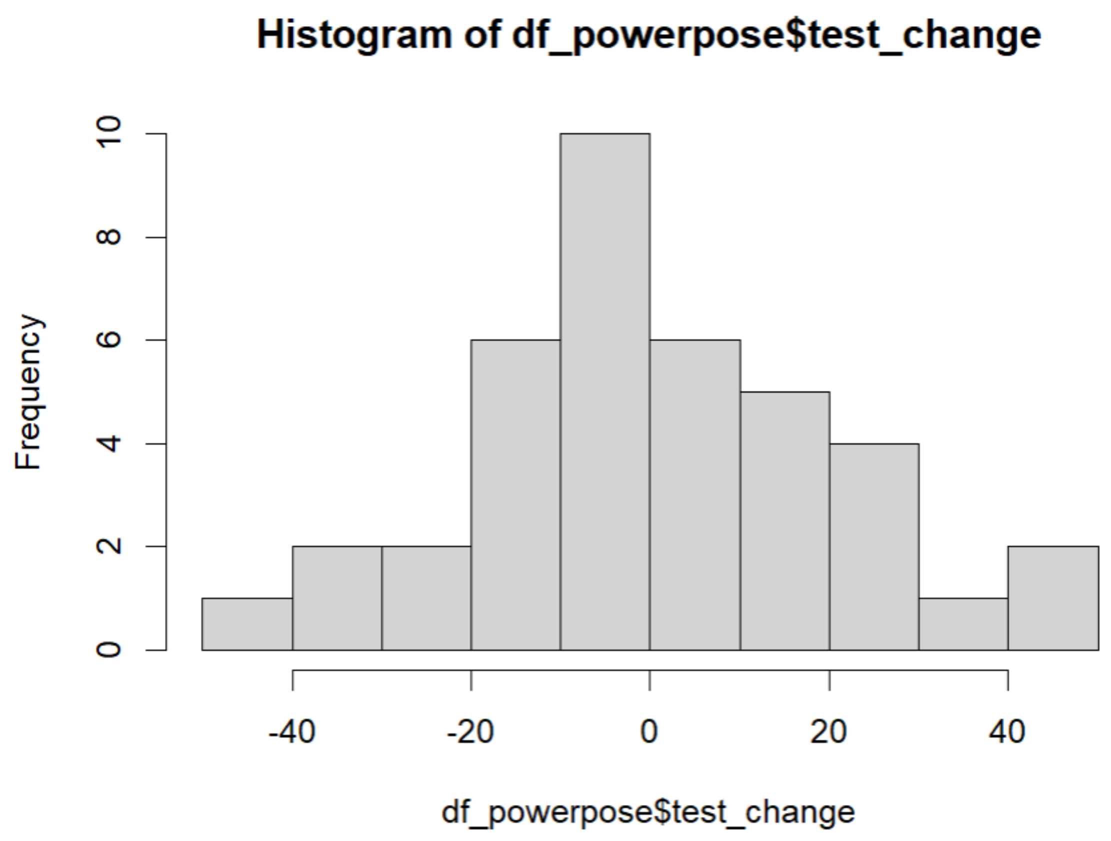
Let’s load the dataset df_powerpose.csv and see the dataset.

```
df_powerpose <- read.table("df_powerpose.csv", header = TRUE, sep = ",")
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

```

We are looking at testosterone levels measured before (testm1) and after (testm2) a treatment. The main thing we're interested in is the type of treatment each person received - either a **high-power pose** or a **low-power pose**.

To evaluate the effect of treatment, we will analyze the **change in testosterone levels** (i.e., the difference between testm2 and testm1) and examine how this change is influenced by the treatment condition (hptreat).

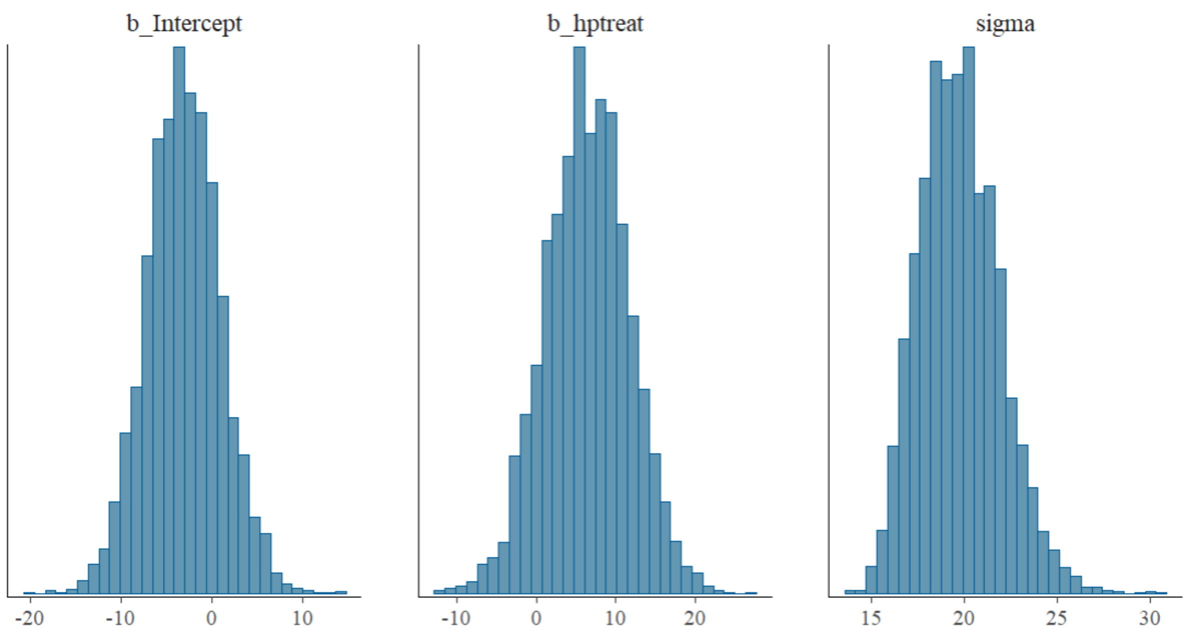
```
df_powerpose$test_change <- df_powerpose$testm2 - df_powerpose$testm1
head(df_powerpose)

## X id hptreat female age testm1  testm2 test_change
## 1 2 29    High   Male  19 38.725  62.375   23.650002
## 2 3 30     Low Female  20 32.770  29.235   -3.534999
## 3 4 31    High Female  20 32.320  27.510   -4.810000
## 4 5 32     Low Female  18 17.995  28.655   10.660000
## 5 7 34     Low Female  21 73.580  44.670  -28.910004
## 6 8 35    High Female  20 80.695 105.485   24.790000

hist(df_powerpose$test_change)

```



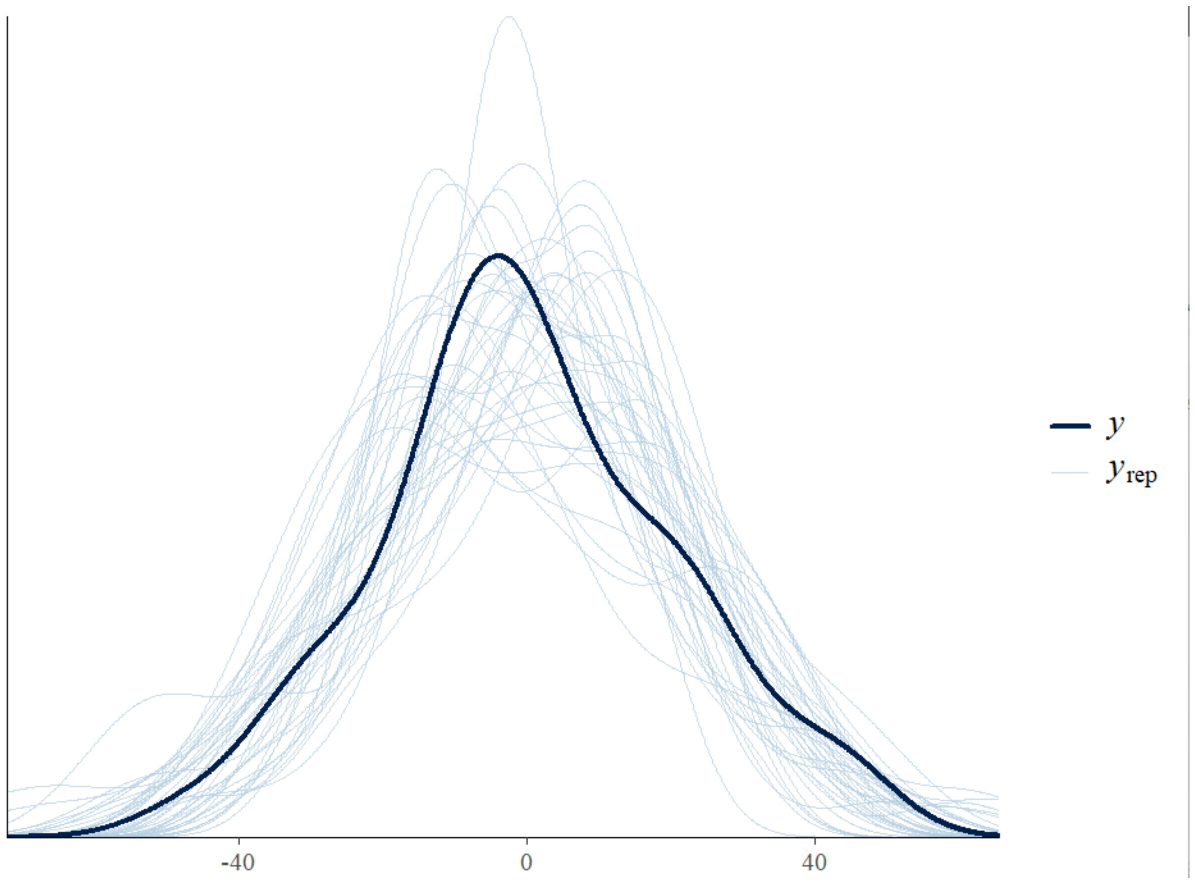
```r
#####################

# Prior assumptions


# test_change ~ N(mu, sigma)
# mu ~ alpha + beta*tptreat
# alpha ~ N(0, 10)
# beta ~ N(0, 10)
# sigma ~ N(0, 10)

install.packages("brms")
install.packages("rstan")
install.packages("rstudioapi")
library(brms)
library(rstan)
library(bayesplot)

df_powerpose$hptreat <- ifelse(df_powerpose$hptreat == "High", 1, 0)

# priors
priors <- c(prior(normal(0, 10), class = Intercept),
            prior(normal(0, 10), class = b, coef = hptreat),
            prior(normal(0, 10), class = sigma))

m1 <- brm(formula = test_change ~ 1+hptreat,
          data = df_powerpose,
          prior = priors,
          family = gaussian(),
          chains = 4, cores = 4,
          iter = 2000, warmup = 1000)

```

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

summary(m1)

## Family: gaussian
##  Links: mu = identity; sigma = identity
## Formula: test_change ~ 1 + hptreat
##   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    -3.11      4.09   -11.04     5.09 1.00     4115     2678
## hptreat       6.38      5.30    -3.91    16.69 1.00     3878     2639
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    19.74      2.17    16.00    24.39 1.00     3481     2663
##
## 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).

```

```
# plotting the histogram
mcmc_hist(m1, pars = c("b_Intercept", "b_hptreat", "sigma"))

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

```



```
# posterior prediction check
pp_check(m1, ndraws = 39, type = "dens_overlay")

```



Part 2: Poisson regression models and hypothesis testing

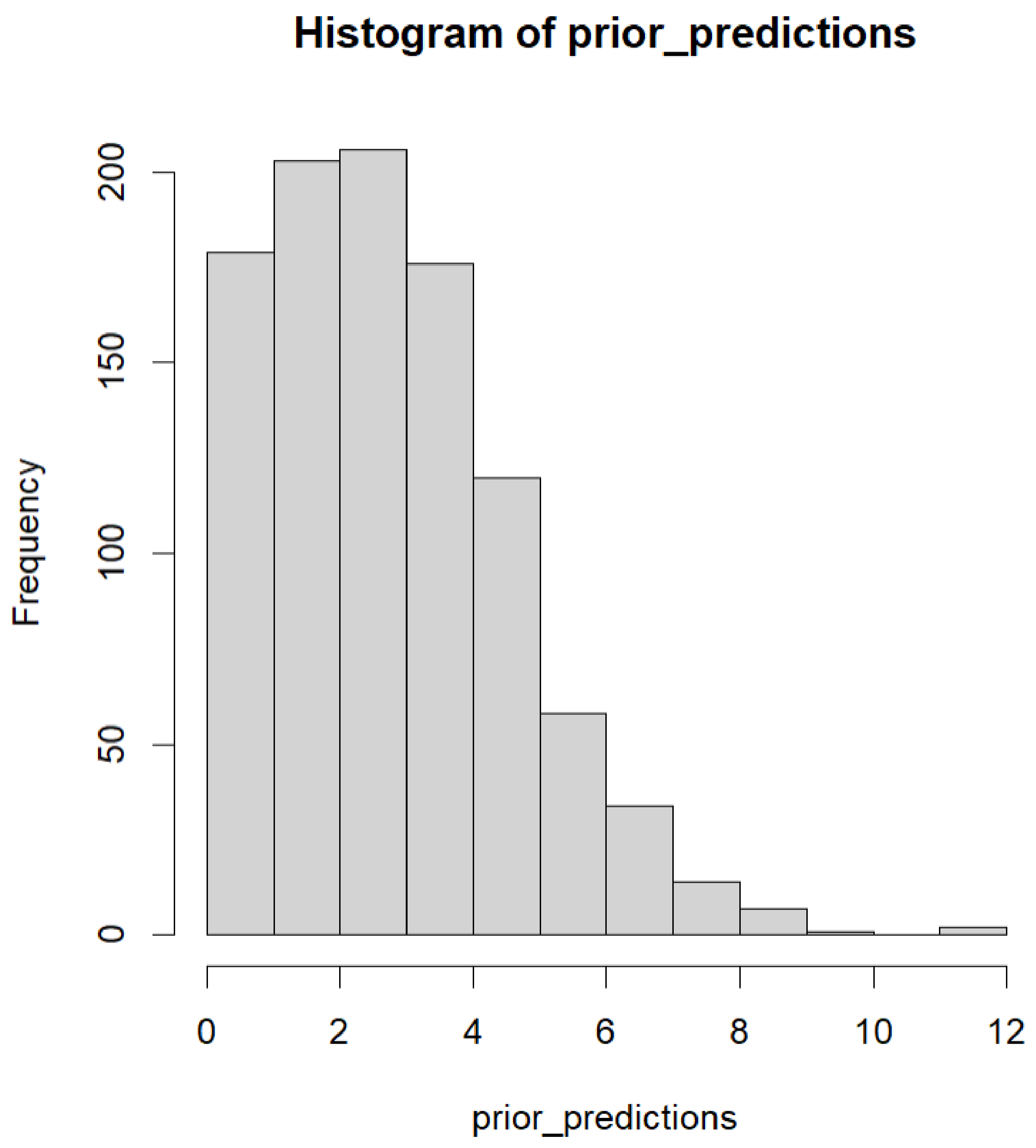
2.1

```r
######################

# Part 2


### 2.1
crossing_model <- function(len, alpha, beta) {
  lambda <- exp(alpha + len*beta)
  N <- rpois(1, lambda)
  return(N)
}

a <- crossing_model(10, 0.15, 0.25)
a

## [1] 15

```

2.2

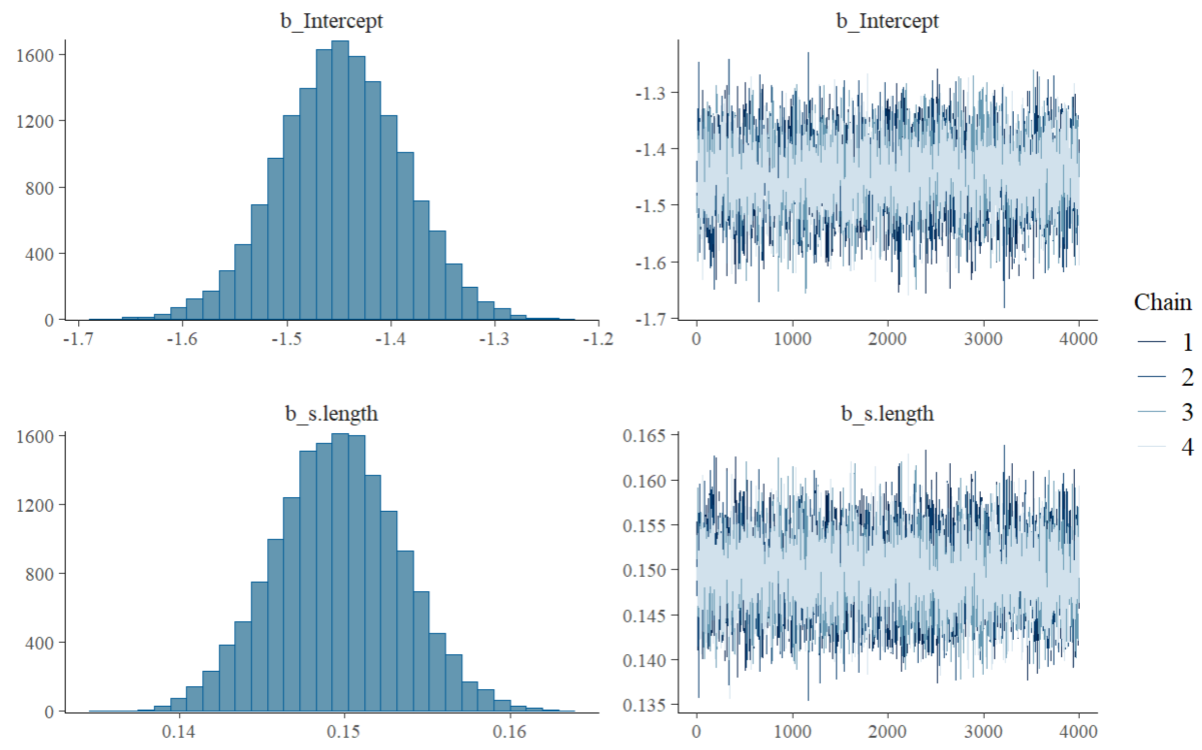
```r
### 2.2
# alpha ~ Normal_lb=0 (0.15, 0.1)
# beta ~ Normal_lb=0 (0.25, 0.05)

library(truncnorm)
alpha_prior <- rtruncnorm(1000, a=0, mean=0.15, sd=0.1)
```

```r
beta_prior <- rtruncnorm(1000, a=0, mean=0.25, sd=0.05)

len <- 4
lambda <- exp(alpha_prior + len*beta_prior)
prior_predictions <- rpois(1000, lambda)
summary(prior_predictions)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.000   2.000   3.000   3.253   4.000  12.000

hist(prior_predictions)

```

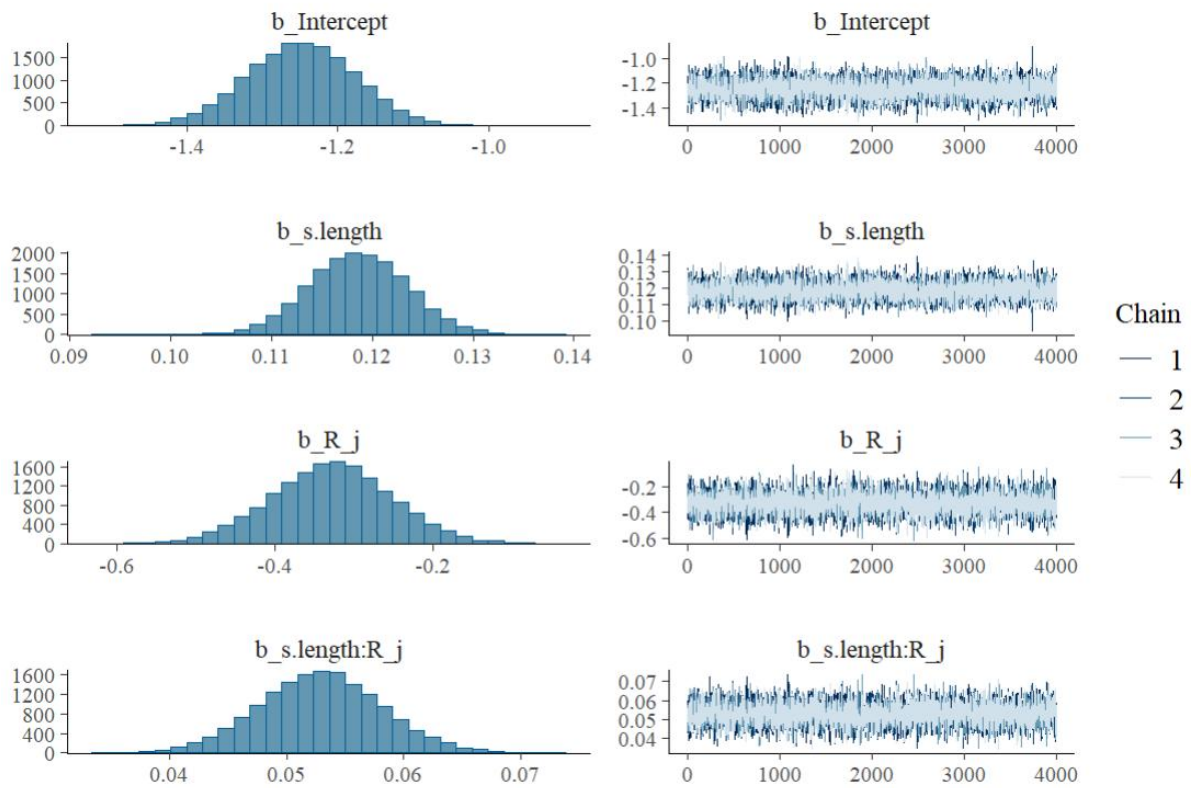


2.3

```
### 2.3

getwd()
df_crossings <- read.table("crossings.csv", header = TRUE, sep=",")
head(df_crossings)


##   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

df_crossings$R_j <- ifelse(df_crossings$Language == "German", 1, 0)
len_ij <- df_crossings$s.length

# model 1
priors <- c(prior(normal(0.15, 0.1), class = Intercept),
            prior(normal(0, 0.15), class = b, coef = s.length))

model_1 <- brm(formula = nCross ~ 1 + s.length,
               data = df_crossings,
               prior = priors,
               family = poisson(link = "log"),
               chains = 4, cores = 4,
               iter = 8000, warmup = 4000)

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

summary(model_1)

##  Family: poisson
##   Links: mu = log
## Formula: nCross ~ 1 + s.length
##    Data: df_crossings (Number of observations: 1900)
##   Draws: 4 chains, each with iter = 8000; warmup = 4000; thin = 1;
##          total post-warmup draws = 16000
##
## Regression Coefficients:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    -1.45      0.06    -1.57    -1.34 1.00     5081     6063
## s.length      0.15      0.00     0.14     0.16 1.00     5756     7403
##
## 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).

library(ggplot2)
plot(model_1)

```

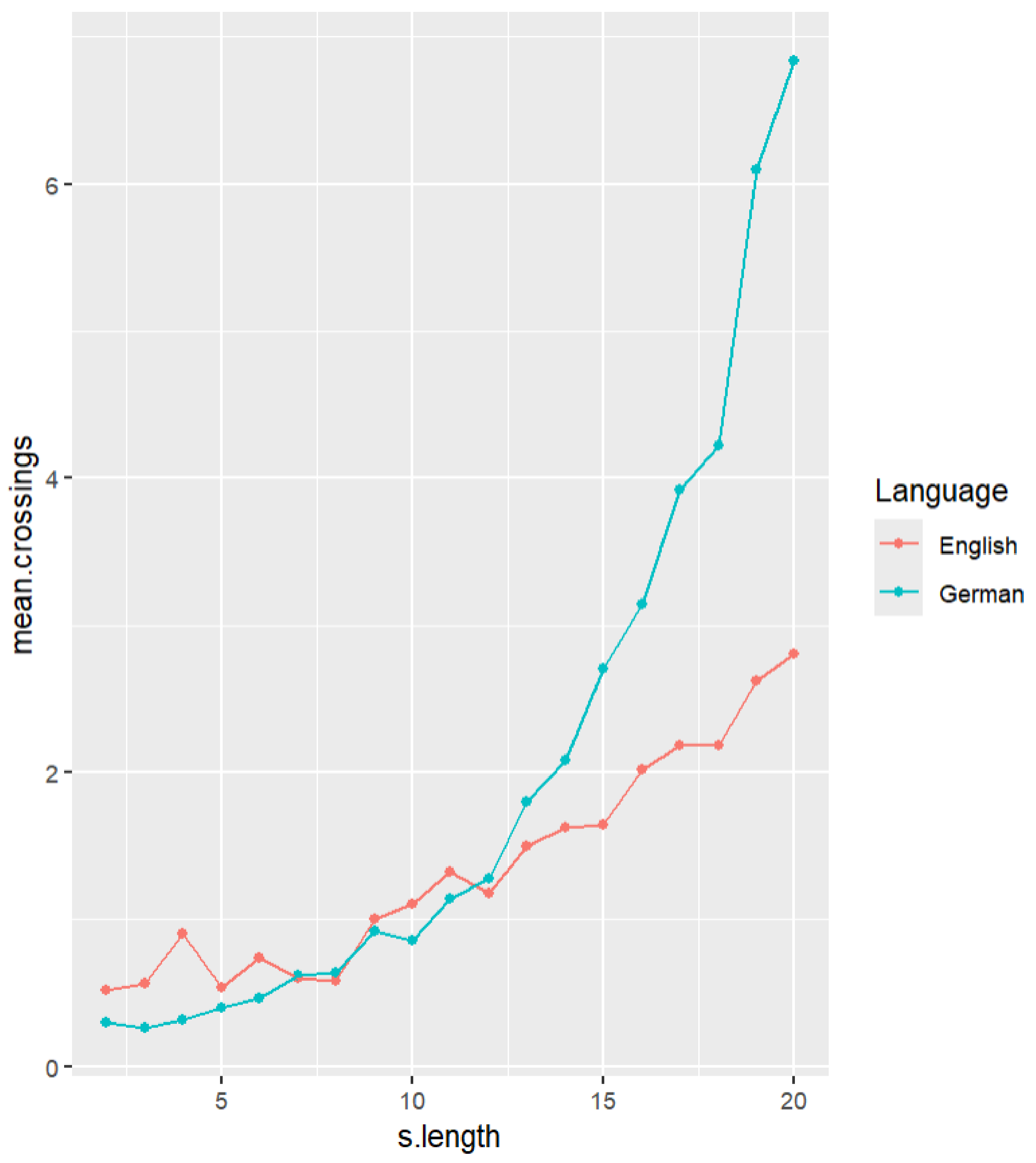


```
# model 2

priors <- c(prior(normal(0.15, 0.1), class = Intercept),
            prior(normal(0, 0.15), class = b, coef = s.length),
            prior(normal(0.15, 0.1), class = b, coef = R_j),
            prior(normal(0.15, 0.1), class = b, coef = s.length:R_j))

model_2 <- brm(formula = nCross ~ 1 + s.length + R_j + s.length*R_j,
               data = df_crossings,
               prior = priors,
               family = poisson(link = "log"),
               chain = 4, cores = 4,
               iter = 8000, warmup = 4000)

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

summary(model_2)

##  Family: poisson
##   Links: mu = log
## Formula: nCross ~ 1 + s.length + R_j + s.length * R_j
##    Data: df_crossings (Number of observations: 1900)
##   Draws: 4 chains, each with iter = 8000; warmup = 4000; thin = 1;
##          total post-warmup draws = 16000
##
## Regression Coefficients:
##              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept       -1.25      0.07    -1.39    -1.11 1.00     6086     7387
## s.length         0.12      0.00     0.11     0.13 1.00     6198     8203
## R_j             -0.33      0.08    -0.48    -0.18 1.00     6203     7081
## s.length:R_j     0.05      0.01     0.04     0.06 1.00     5983     6724
##
## 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(model_2)
```



2.4

```
### 2.4

library(plyr)
library(dplyr)

## Attaching package: ‘dplyr’
##
## The following objects are masked from ‘package:plyr’:
##
##     arrange, count, desc, failwith, id,
##     mutate, rename, summarise, summarize
##
## The following objects are masked from ‘package:stats’:
##
##     filter, lag
##
## The following objects are masked from ‘package:base’:
##
##     intersect, setdiff, setequal, union

# Visualize average rate of crossings
observed <- read.table("crossings.csv", header = , sep = ",")

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.

```



```r
# Code/center the predictors
observed$s.length <- observed$s.length - mean(observed$s.length)
observed$lang <- ifelse(observed$Language == "German", 0, 1)

# These two vector will store log predictive densities
# in each fold

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

for (k in 1:5) {
  # prepare test data and training data
  y_test <- sample_n(untested, size = nrow(observed)/5)
  y_train <- setdiff(observed, y_test)
  untested <- setdiff(untested, y_test)

  fit.m1 <- brm(nCross ~ 1 + s.length,
                data = y_train,
                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 = y_train,
                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)

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

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

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

  }

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

```

```r
# predective accuracy of model M1
elpd.m1 <- sum(lpds.m1)
elpd.m1

## [1] -2817.517


# predective accuracy of model M2
elpd.m2 <- sum(lpds.m2)
elpd.m2

## [1] -2683.562

# Evidence in favour of M2 over M1
difference_elpd <- elpd.m2-elpd.m1
difference_elpd

## [1] 133.9556

```