CGS698C - Assignment 4

Sahil Tomar (210898)

2024-07-03

Part 1 : Power posing and testosterone

```
df_powerpose <- read.table("df_powerpose.csv", header=T, sep=",")
flextable(head(df_powerpose))
```

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

The data set shows the testosterone levels of 39 different individuals, before `(testm1)` and after `(testm2)` treatment, where treatment refers to each individual being assigned to a high power pose or a low power pose `(hptreat)`.

The research hypothesis is that on average, assigning a subject a high power pose vs. a low power pose will lead to higher testosterone levels after treatment.

```
df_powerpose <- df_powerpose %>%
  mutate(
    hptreat_binary = ifelse(hptreat == "High", 1, 0),
    change_testosterone = testm2 - testm1
  )

mfit <- brm(
  formula = change_testosterone ~ hptreat_binary,
  data = df_powerpose,
  family = gaussian(),
  chains = 4, cores = 4,
  iter = 4000, warmup = 1000,
  verbose = FALSE
)
```

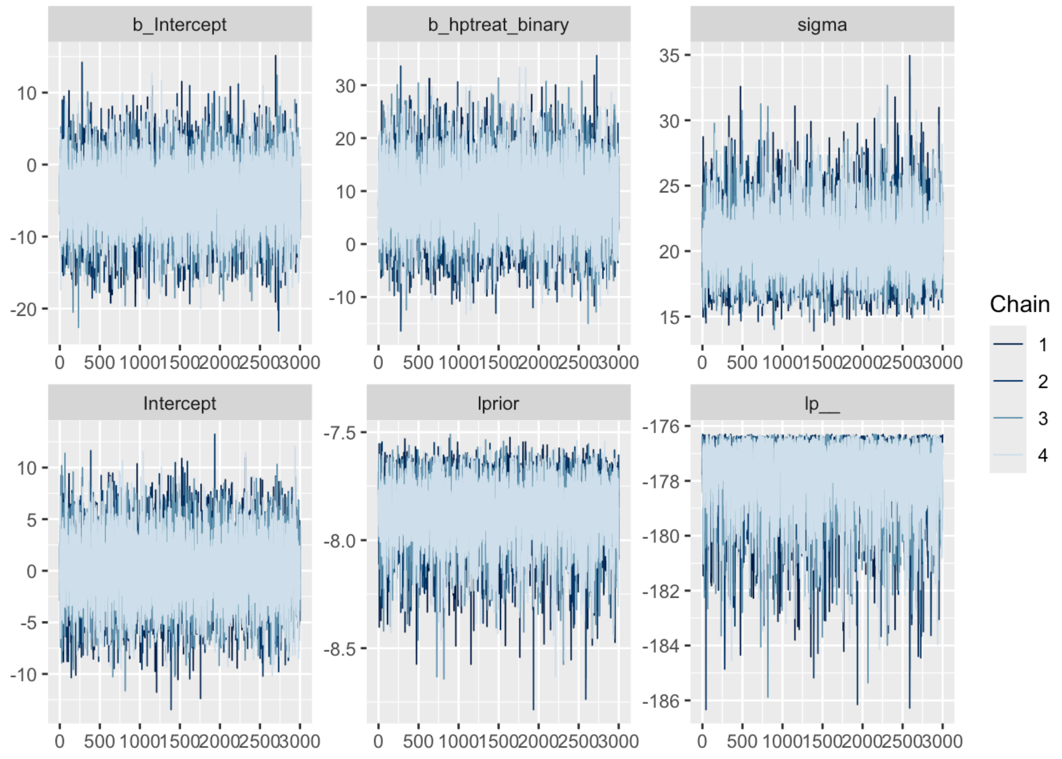
```
summary(mfit)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: change_testosterone ~ hptreat_binary
##    Data: df_powerpose (Number of observations: 39)
##   Draws: 4 chains, each with iter = 4000; warmup = 1000; thin = 1;
##          total post-warmup draws = 12000
##
## Regression Coefficients:
##                Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept         -4.49      4.68   -13.79     4.50 1.00    11145     8478
## hptreat_binary     8.89      6.59    -3.99    21.66 1.00    11760     8564
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    20.55      2.41    16.46    25.83 1.00     9095     8312
##
## 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).
```

The above summary provides the value of the hptreat_binary coefficient (β) which is the measure of the effect of the variable that encodes the change in testosterone.

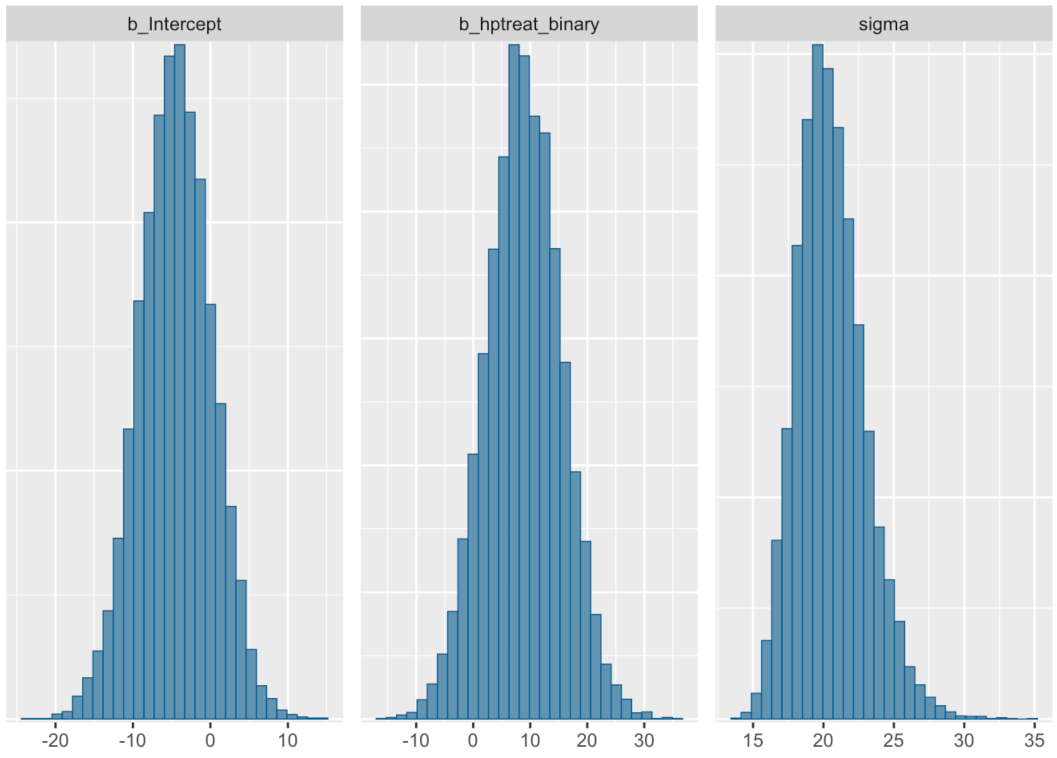
The α is mostly in the negative region while β is mostly in the positive regions, this suggests that testosterone tends to increase after a high power pose.

```
mcmc_trace(mfit)
```



```
mcmc_hist(mfit, pars=c("b_Intercept", "b_hptreat_binary", "sigma"))
```

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



Prior Sensitivity Analysis

Using the above summary knowledge, these priors are some priors one can check for sensitivity -

```
# Define prior specifications for sensitivity analysis
prior_specs <- list(
  list(alpha = "normal(-5, 1)", beta = "normal(8, 1)"),
  list(alpha = "normal(-4, 1)", beta = "normal(8, 1)"),
  list(alpha = "normal(-5, 2)", beta = "normal(8, 1)"),
  list(alpha = "normal(-4, 2)", beta = "normal(8, 1)"),
  list(alpha = "normal(-5, 1)", beta = "normal(9, 1)"),
  list(alpha = "normal(-4, 1)", beta = "normal(9, 1)"),
  list(alpha = "normal(-5, 2)", beta = "normal(9, 1)"),
  list(alpha = "normal(-4, 2)", beta = "normal(9, 1)")
)

# Initialize an empty data frame to store sensitivity results
df.sensitivity <- data.frame(matrix(nrow = 0, ncol = 5))
colnames(df.sensitivity) <- c("prior_alpha", "prior_beta", "mean_beta", "q.lower", "q.upper")

# Fit the model and store results for each prior specification
for (spec in prior_specs) {
  # Update priors for alpha and beta
  priors <- c(
    set_prior(spec$alpha, class = "Intercept"),
    set_prior(spec$beta, class = "b", coef = "hptreat_binary"),
    set_prior("normal(0, 10)", class = "sigma")  # Keep the prior for sigma unchanged
  )

  # Fit the model
  mfit <- brm(
    formula = change_testosterone ~ hptreat_binary,
    data = df_powerpose,
    family = gaussian(),
    prior = priors,
    chains = 4,
    cores = 4,
    iter = 2000,
    warmup = 1000,
    seed = 123  # Set a seed for reproducibility
  )

  # Extract posterior samples of beta (hptreat_binary)
  post_samples <- posterior_samples(mfit)

  # Calculate mean, lower, and upper quantiles of beta (hptreat_binary)
  mean_beta <- mean(post_samples$b_hptreat_binary)
  lower_quantile <- quantile(post_samples$b_hptreat_binary, probs = 0.025)
  upper_quantile <- quantile(post_samples$b_hptreat_binary, probs = 0.975)

  # Store results in df.sensitivity
  df.sensitivity[nrow(df.sensitivity) + 1, ] <- c(
    spec$alpha,
    spec$beta,
    mean_beta,
    lower_quantile,
    upper_quantile
  )

  # Save the model for each prior specification if needed
}
```

```
flextable(df.sensitivity)
```

prior_alpha	prior_beta	mean_beta	q.lower	q.upper
normal(-5, 1)	normal(8, 1)	8.03562027075994	6.07380653329301	9.99240813270475
normal(-4, 1)	normal(8, 1)	8.02379996721376	6.11090050841724	9.91721994045697
normal(-5,	normal(8,	8.02669329520304	6.11467300835655	9.9432372410656

prior_alpha	prior_beta	mean_beta	q.lower	q.upper
2)	1)			
normal(-4, 2)	normal(8, 1)	8.02424678426533	6.0597824934006	9.90882312485173
normal(-5, 1)	normal(9, 1)	8.98046596349485	7.06759186473789	10.8960662359089
normal(-4, 1)	normal(9, 1)	8.99863379987108	7.09718310024839	10.8531113415611
normal(-5, 2)	normal(9, 1)	8.97507838553259	7.05452706582347	10.9508412809515
normal(-4, 2)	normal(9, 1)	9.00312282250039	7.0918239009274	10.8945481320449

Part 2 : Poisson regression models and hypothesis testing

Exercise 2.1 -

Implement the model in R or Python such that the function gives the number of crossings as the outcome, and takes sentence length, α, and β as its arguments.

We are given:

- The number of crossing dependencies in a sentence can be given by a Poisson distribution - N_i ~ $Poisson(\lambda_i)$
- where N_i is the number of crossing dependencies in the sentence i
- λ_i is rate parameter indicating the expected rate of crossing dependencies in the sentence i, such that $log\lambda_i = \alpha + \beta L_i$
- where L_i is the length of the sentence i, α is the expected rate of crossings in a sentence of average length and β is the change in rate of crossings as a function of sentence length.

```
calculate_crossings <- function(sentence_length, alpha, beta) {
  lambda_i <- exp(alpha + beta * sentence_length)
  crossings <- rpois(1, lambda_i)
  return(crossings)
}
```

Above function can be used to get the number of crossings as the outcome given the sentence length, α, and β as its arguments.

Exercise 2.2 -

Generate prior predictions of the model for sentences of length 4 under the following prior assumptions -

α ~ $Normal_{lb=0}(0.15, 0.1)$ and β ~ $Normal_{lb=0}(0.25, 0.05)$

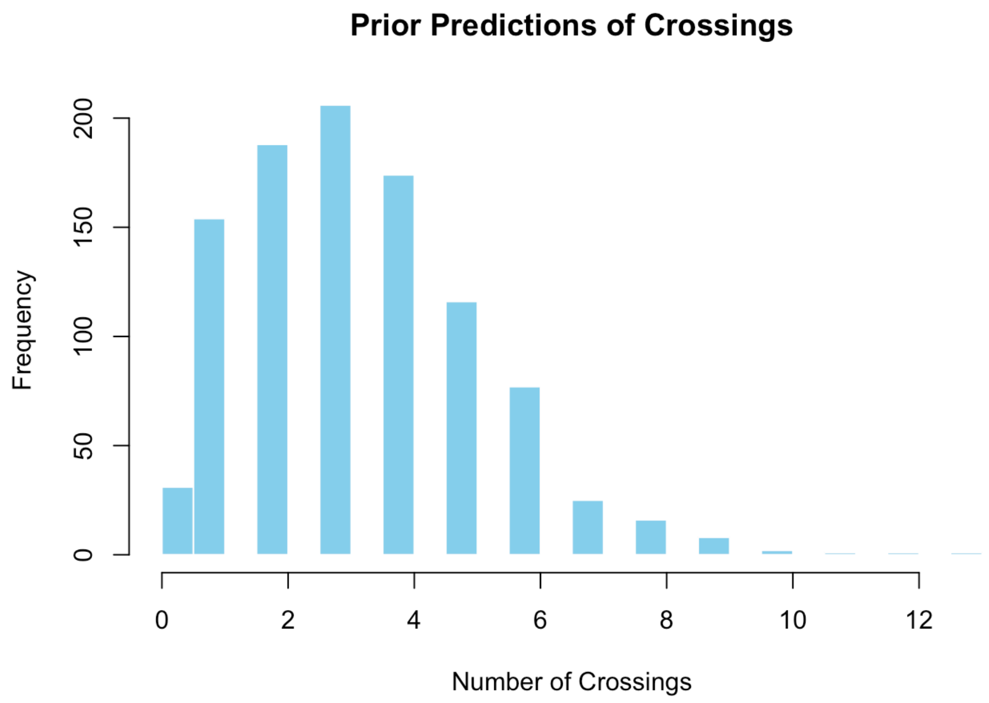
```
# Sample alpha from truncated Normal(0.15, 0.1)
alpha <- rtruncnorm(1000, a = 0, b = Inf, mean = 0.15, sd = 0.1)

# Sample beta from truncated Normal(0.25, 0.05)
beta <- rtruncnorm(1000, a = 0, b = Inf, mean = 0.25, sd = 0.05)

sentence_length <- 4

prior_crossings <- numeric(length(alpha))
for (i in 1:length(alpha)) {
  prior_crossings[i] <- calculate_crossings(sentence_length, alpha[i], beta[i])
}

# Plotting histogram of prior crossings
hist(prior_crossings, breaks = 20, col = "skyblue", border = "white",
     xlab = "Number of Crossings", main = "Prior Predictions of Crossings")
```



Exercise 2.3 -

Consider a dataset of crossing dependencies from English and German corpora, "crossing.csv". This dataset contains number of crossings for each sentence from each language. Fit the following two models, M_1 and M_2, to the given data.

Model $M1$:

- Assumption -

  The rate of crossings is only a function of sentence length and it remains exactly the same in English and German.

- Regression - $N_{i,j}$ ~ $Poisson(\lambda_{i,j})$ where $N_{i,j}$ is the number of crossing dependencies in sentence i in language j; $\lambda_{i,j}$ is rate parameter indicating the expected rate of crossing dependencies in sentence i in language j, such that $log(\lambda_{i,j}) = \alpha + \beta L_{ij}$ where $L_{i,j}$ is the length of sentence i of language j.

Model $M2$:

- Assumption -

  As sentence length increases, the number crossings grows at a different rate in English vs. German.

- Regression - $N_{i,j}$ ~ $Poisson(\lambda_{i,j})$ where $N_{i,j}$ is the number of crossing dependencies in sentence i in language j; $\lambda_{i,j}$ is rate parameter indicating the expected rate of crossing dependencies in sentence i in language j, such that $log(\lambda_{i,j}) = \alpha + \beta L_{ij} + \beta_{language} R_j + \beta_{interact} L_{i,j} * R_j$ where $L_{i,j}$ is the length of sentence i of language j, R_j is the indicator variable such that $R_j = 0$ if the language is English and $R_j = 1$ if the language is German.

Priors :

- α ~ $Normal(0.15, 0.1)$
- β ~ $Normal(0, 0.15)$
- $\beta_{language}$ ~ $Normal(0, 0.15)$
- $\beta_{interact}$ ~ $Normal(0, 0.15)$

```
observed <- read.table("crossings.csv", header = T, sep = ",")
flextable(head(observed))
```

Language	s.id	s.length	nCross
German	1	2	0
German	2	2	1
German	3	2	0
German	4	2	0

Language	s.id	s.length	nCross
German	5	2	2
German	6	2	1

```
nrow(observed)
```

```
## [1] 1900
```

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

```
m1.fit <- brm(
  nCross ~ 1 + s.length,
  family = poisson(link = "log"),
  data = observed,
  prior = c(prior(normal(0.15, 0.1), class = Intercept),
            prior(normal(0, 0.15), class = b)),
  cores = 4,
  verbose = FALSE
)
```

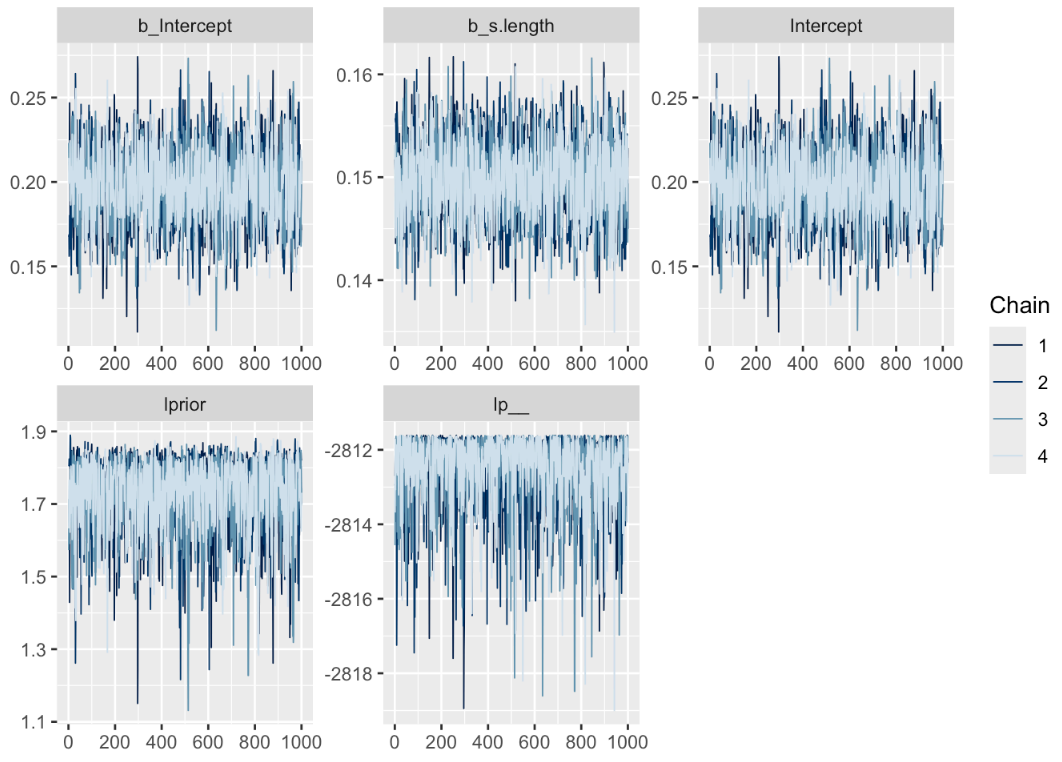
```
summary(m1.fit)
```

```
##  Family: poisson
##   Links: mu = log
## Formula: nCross ~ 1 + s.length
##    Data: observed (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     0.20      0.02     0.15     0.24 1.00     1394     2009
## s.length      0.15      0.00     0.14     0.16 1.00     1627     2224
##
## 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).
```

After fitting model M_1 using the data we get α - 0.1968321, 0.0218648, 0.1549425, 0.2393788

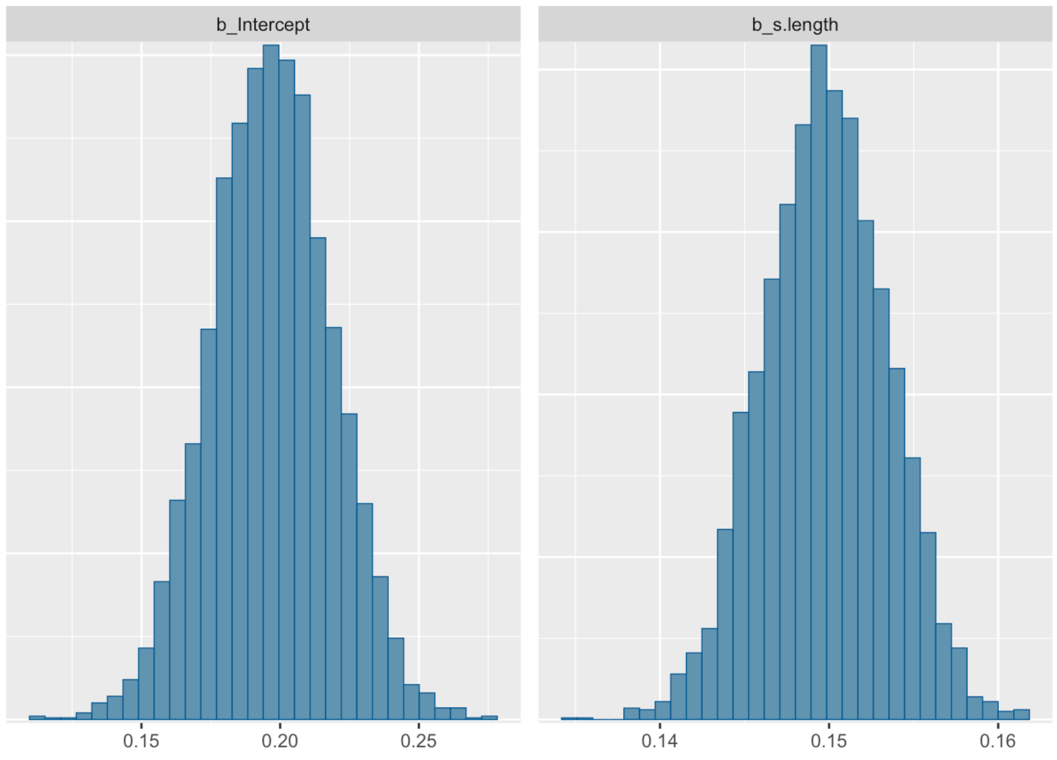
After fitting model M_1 using the data we get β - 0.1497753, 0.0036893, 0.1425566, 0.1568218

```
mcmc_trace(m1.fit)
```



```
mcmc_hist(m1.fit, pars = c("b_Intercept", "b_s.length"))
```

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



```
m2.fit <- brm(
  nCross ~ 1 + s.length + lang + s.length*lang,
  family = poisson(link = "log"),
  data = observed,
  prior = c(prior(normal(0.15, 0.1), class = Intercept),
            prior(normal(0, 0.15), class = b)),
  cores = 4,
  verbose = FALSE
)
```

```
summary(m2.fit)
```

```
##  Family: poisson
##   Links: mu = log
## Formula: nCross ~ 1 + s.length + lang + s.length * lang
##    Data: observed (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          0.16      0.03     0.10     0.22 1.00     2058     2155
## s.length           0.10      0.01     0.09     0.11 1.00     2053     2048
## lang               0.02      0.05    -0.07     0.12 1.00     2033     2105
## s.length:lang      0.10      0.01     0.08     0.11 1.00     1976     2063
##
## 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).
```

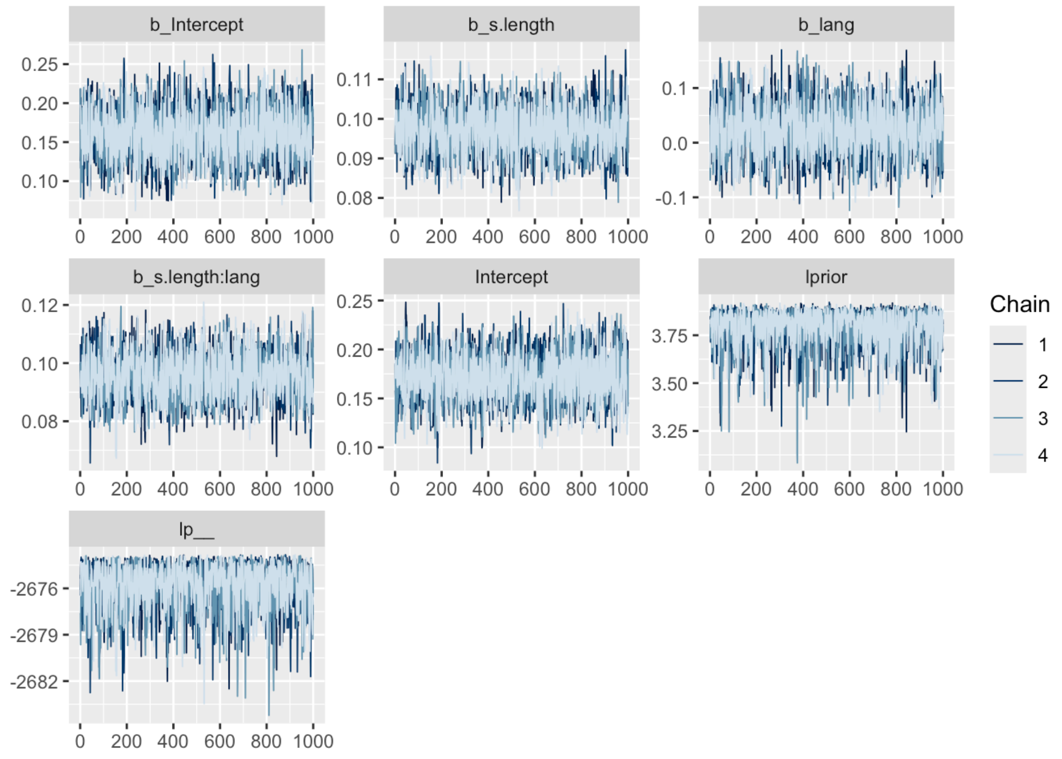
After fitting model M_2 using the data we get α - NA

After fitting model M_2 using the data we get β - NA

After fitting model M_2 using the data we get $\beta_{language}$ - NA

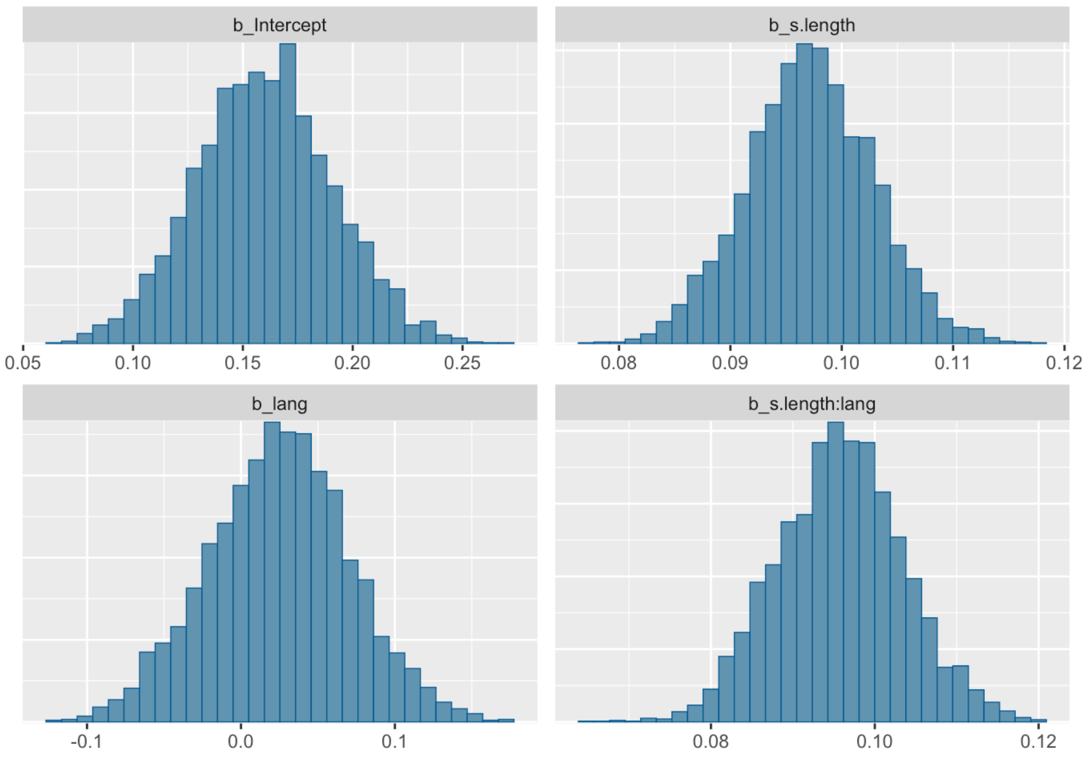
After fitting model M_2 using the data we get $\beta_{interaction}$- NA

```
mcmc_trace(m2.fit)
```



```
mcmc_hist(m2.fit, pars = c("b_Intercept", "b_s.length", "b_lang", "b_s.length:lang"))
```

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

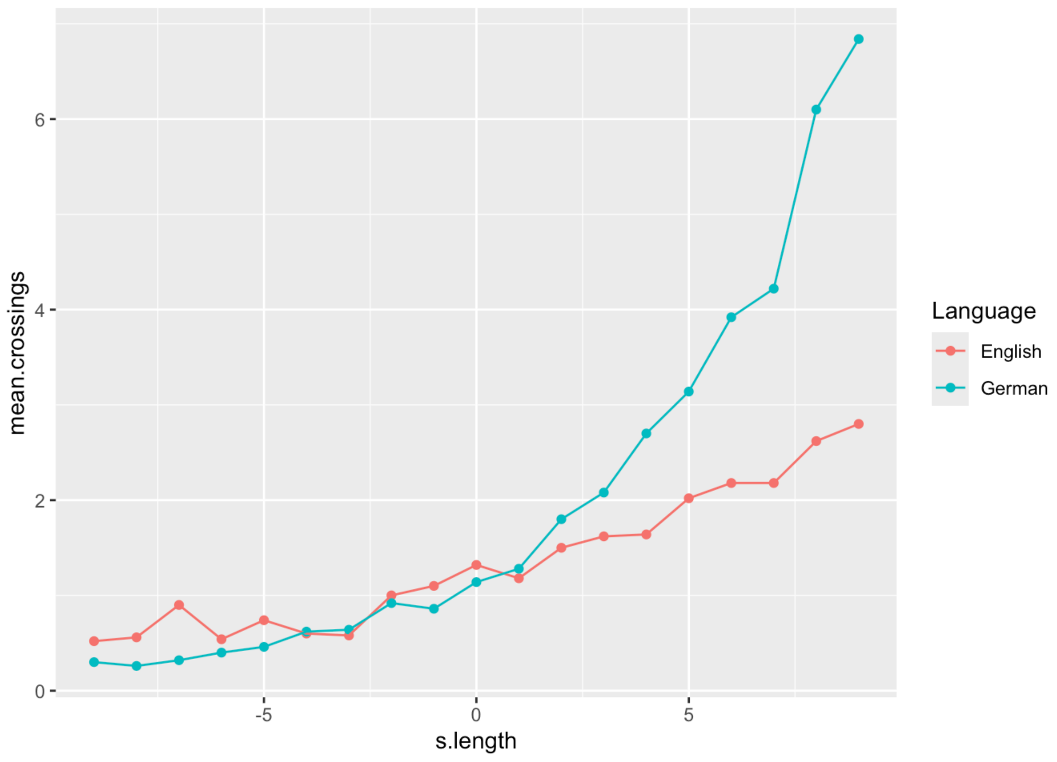


Exercise 2.4 -

Quantify evidence for the models M1 and M2 using k-fold cross-validation.

```
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.
```



```
# These two vectors will store log predictive desnities
# in each fold
lpds.m1 <- c()
lpds.m2 <- c()

untested <- observed

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

  untested <- setdiff(untested,ytest)
# Fit the models M1 and M2 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,
        verbose = FALSE)
  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,
        verbose = FALSE)
  # retrieve posterior samples
  post.m1 <- posterior_samples(fit.m1)
  post.m2 <- posterior_samples(fit.m2)

  # Calculate log pointwise predcitive 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
  }
  lpds.m1 <- c(lpds.m1,lppd.m1)
  lpds.m2 <- c(lpds.m2,lppd.m2)
}
# Predictive accuracy of model M1
elpd.m1 <- sum(lpds.m1)
elpd.m1_SE <- sqrt(5*var(lpds.m1))

# Predictive accuracy of model M2
elpd.m2 <- sum(lpds.m2)
elpd.m2_SE <- sqrt(5*var(lpds.m2))

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

```
elpd.m1
```

```
## [1] -2814.18
```

```
elpd.m1_SE
```

```
## [1] 36.64415
```

```
elpd.m2
```

```
## [1] -2680.777
```

```
elpd.m2_SE
```

```
## [1] 13.41668
```

```
difference_elpd
```

```
## [1] 133.4034
```

Conclusions based on the above evidence :

- Model M_2 has a higher ELPD, indicating better predictive accuracy compared to Model M_1.
- The positive difference `(difference_elpd=134.1365)` favors Model M_2, suggesting that Model M_2 outperforms Model M_1 in terms of predictive accuracy.
- Both models have relatively low standard errors, indicating reasonably precise estimates of their respective ELPD values. However, Model M_2 has a slightly lower standard error, suggesting greater confidence in its ELPD estimate compared to Model M_1.