Assignment 4

Amipriya Anand (220122)

2024-07-02

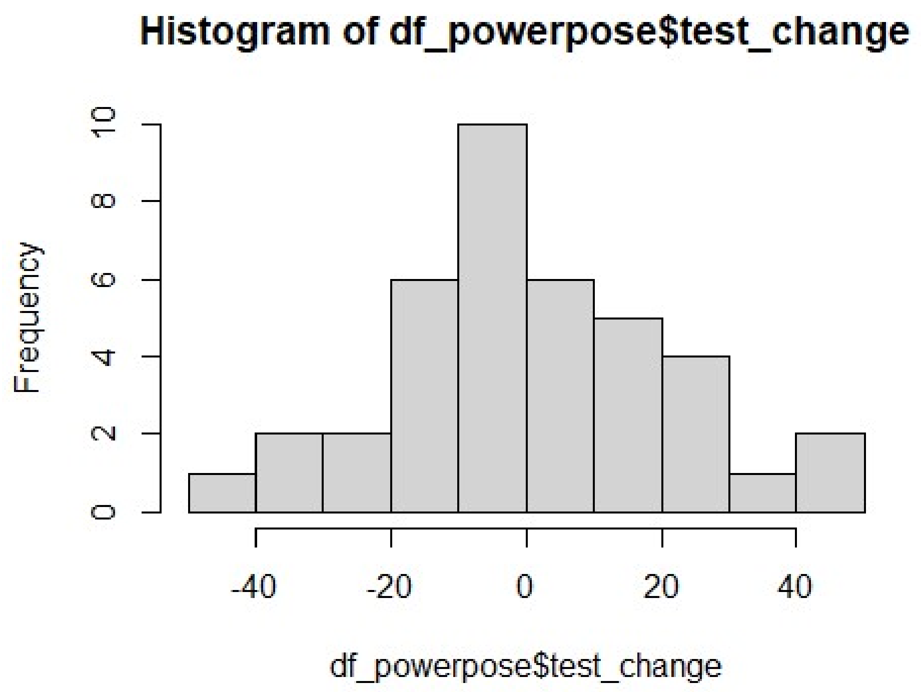
Part 1: A simple linear regression: Power posing and testosterone

First lets load the dataset df_powerpose.csv and see the data set.

```
df_powerpose <- read.table("df_powerpose.csv",header=T,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
```

Now here, the data contains the e testosterone levels before(`testm1`) and after(`testm2`) the treatment. According to the Hypothesis our feature of importance is `hptreat`, what treatment is done high or low pose. So we will consider the effect of treatment on the change in the testosterone levels before and after the treatment.

```
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)
```



Thus our test_change ~ N(expression(mu),espression(sigma)), expression(mu) = expression(alpha)+expression(beta)*hptreat let expression(alpha) ~ N(0,10), expression(beta) ~ N(0,10) and espression(sigma) ~ N(0,10)

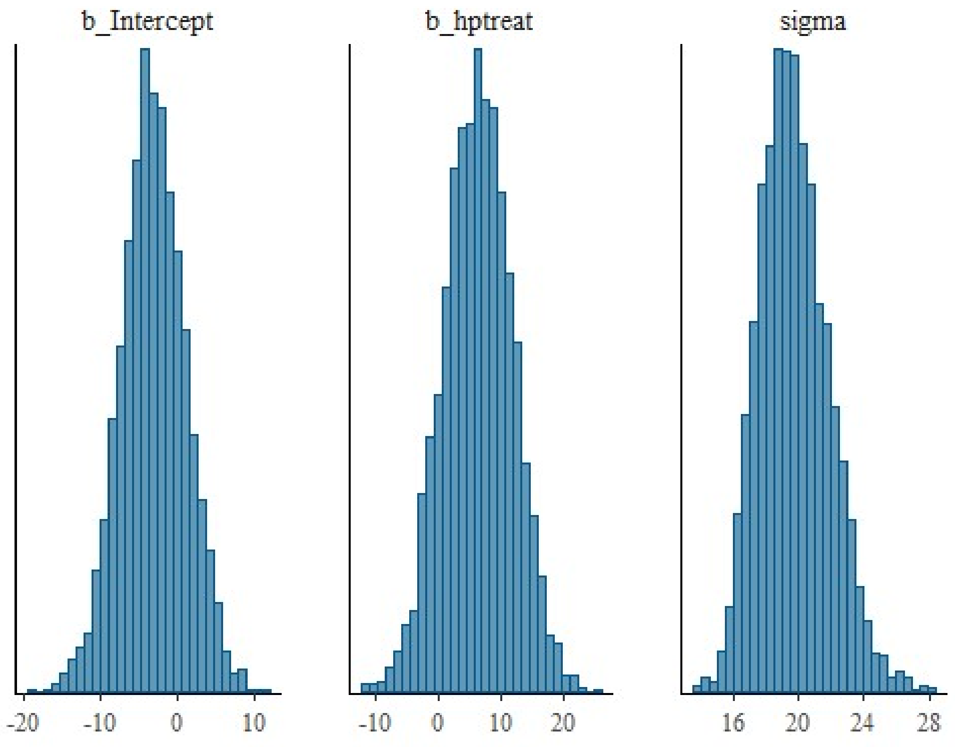
```
library(ggplot2)
library(rstan)

library(brms)

library(bayesplot)

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

# Weakly informative 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)
```

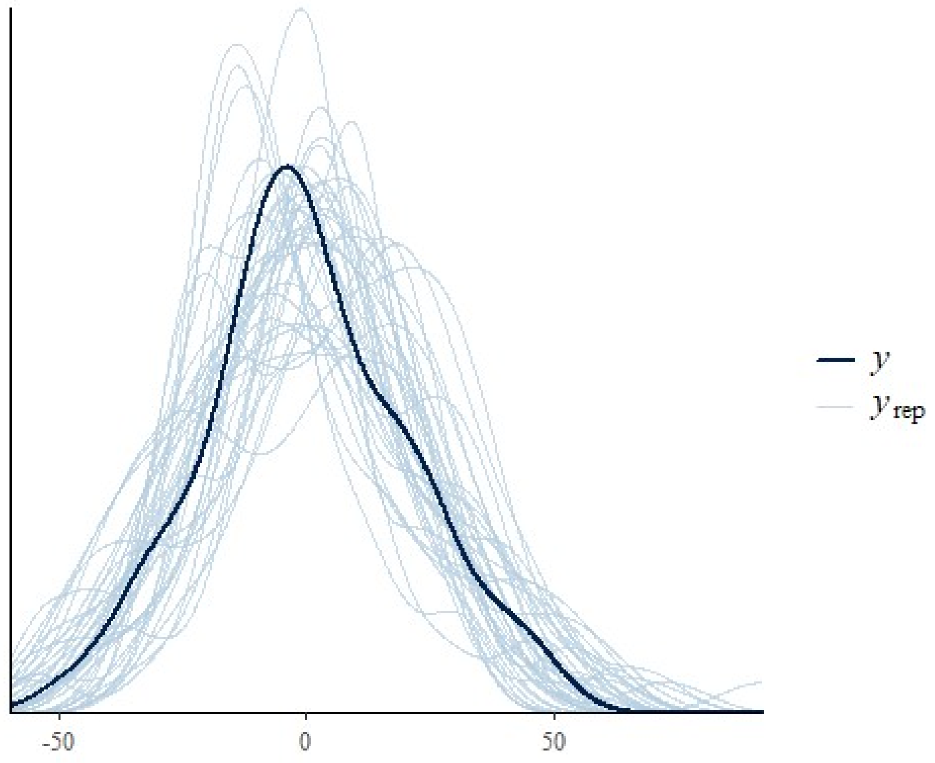
```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: test_change ~ 1 + hptreat
##    Data: df_powerpose (Number of observations: 39)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Regression Coefficients:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    -3.11      4.08   -11.17     4.89 1.00     3350     2902
## hptreat       6.44      5.46    -4.06    16.96 1.00     3491     2690
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    19.71      2.11    16.08    24.16 1.00     3667     2812
##
## 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 predictive check
pp_check(m1,ndraws = 39, type = "dens_overlay")
```



Clearly, it can be seen that as `b_hptreat` which is the beta in our case is lower has 0 included in the 95% credible interval thus the value of beta > 0, this shows that the there is an effect of treatment on the testosterone levels hence the research hypothesis is correct and is consistent with the given data.

Part 2: Poisson regression models and hypothesis testing

As given in the question, the number of crossing dependencies in a sentence can be given by a Poisson distribution Ni ~ Poisson(λi) where Ni is the number of crossing dependencies in the sentence i; λi is rate parameter indicating the expected rate of crossing dependencies in the sentence i, such that log λi = α + βLi where Li is the length of the sentence i, α is the expected rate of crossings in a sentence of average length (say 11) and β is the change in rate of crossings as a function of sentence length.

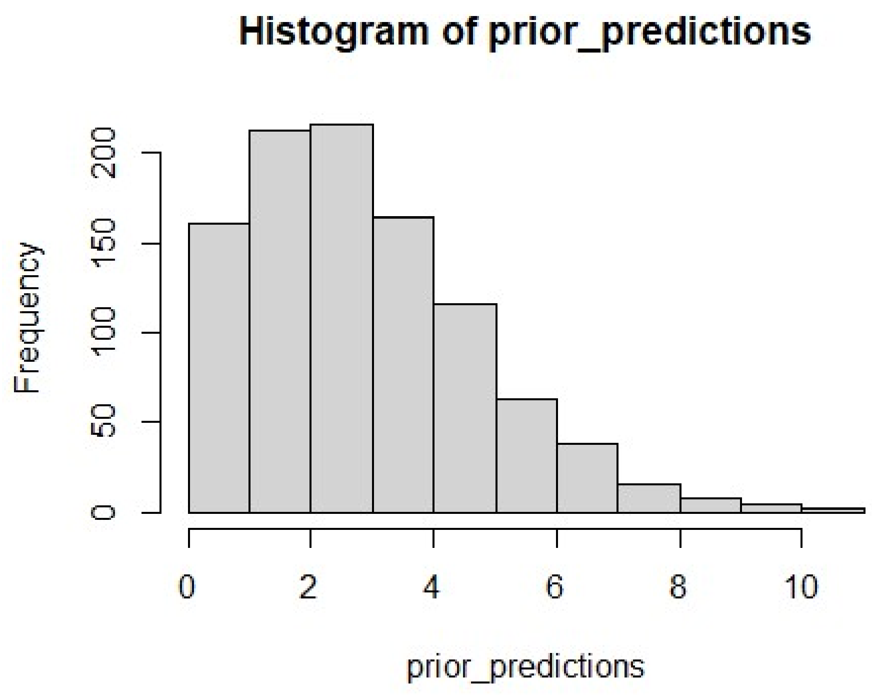
Exercise 2.1: Implement the model in R or Python such that the function gives the number of crossings as the outcome, and takes sentence length, α, and β as its arguments.

```
crossing_model = function(len,alpha,beta)
{
  lambda  = exp(alpha + len*beta)
  N = rpois(1,lambda)    #number of crossings
  return(N)
}
#testing the function
a = crossing_model(10,0.15,0.25)
a

## [1] 13
```

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), β ~ Normal_lb=0 (0.25, 0.05)

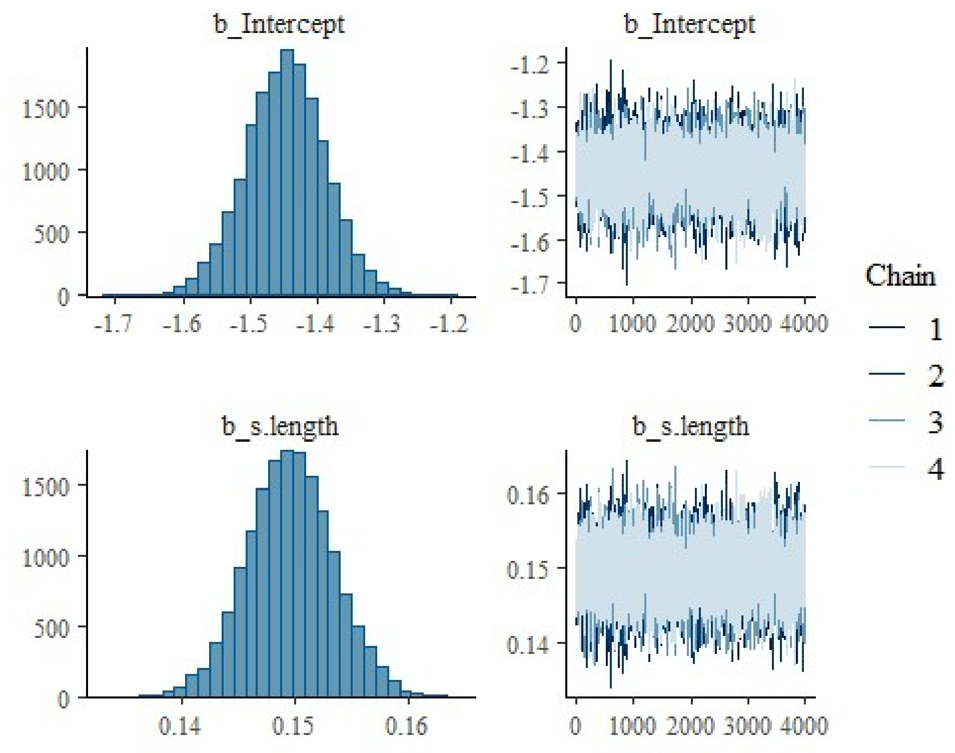
```
alpha_prior = rnorm(1000,0.15,0.1)
beta_prior = rnorm(1000,0.25,0.05)

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   3.333   4.000  11.000

hist(prior_predictions)
```



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, M1 and M2, to the given data.

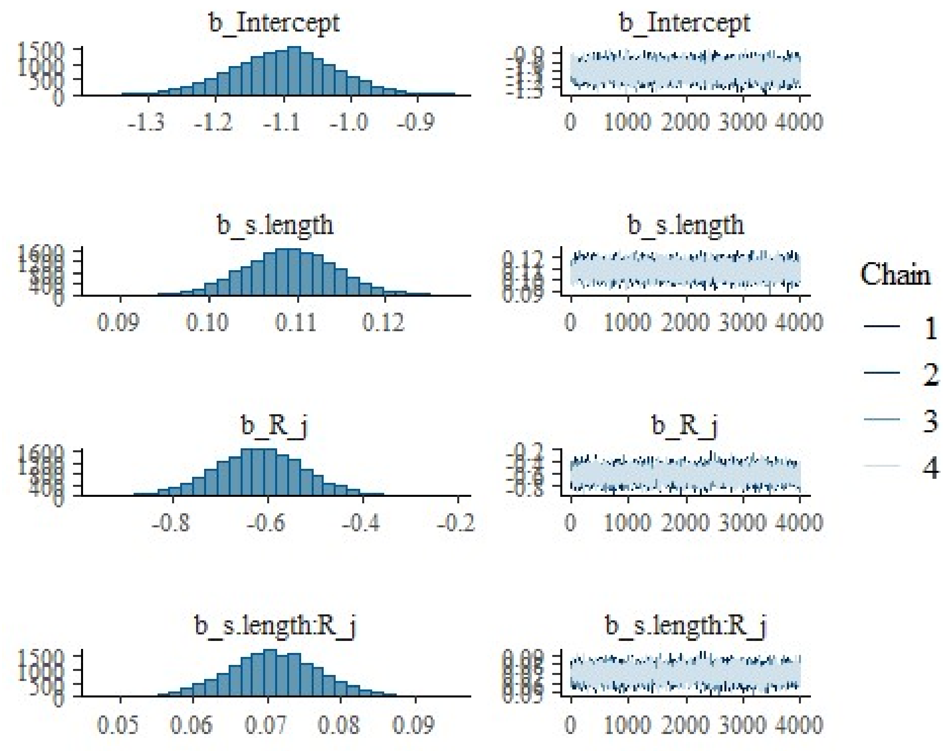
First lets load the dataset, and create another coloumn for the R_j(storing 0 and 1 for english and german language respectively)

```
library(ggplot2)
library(rstan)
library(brms)
df_crossings <- read.table("crossings.csv",header=T,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)
length_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 statistic of model 1
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    -1.33 1.00     4634     6578
## s.length      0.15      0.00     0.14     0.16 1.00     5599     7700
##
## 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 of complete parameters
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(0,0.15),class = b,coef = R_j),
           prior(normal(0,0.15),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"),
              chains = 4,cores = 4,
              iter = 8000, warmup = 4000)

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

#summary statistic of model 1
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         -1.10      0.07    -1.25    -0.95 1.00     6575     7984
## s.length           0.11      0.01     0.10     0.12 1.00     6808     8451
## R_j               -0.62      0.09    -0.81    -0.44 1.00     6132     7319
## s.length:R_j       0.07      0.01     0.06     0.08 1.00     5994     7483
##
## 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 of complete parameters
plot(model_2)
```

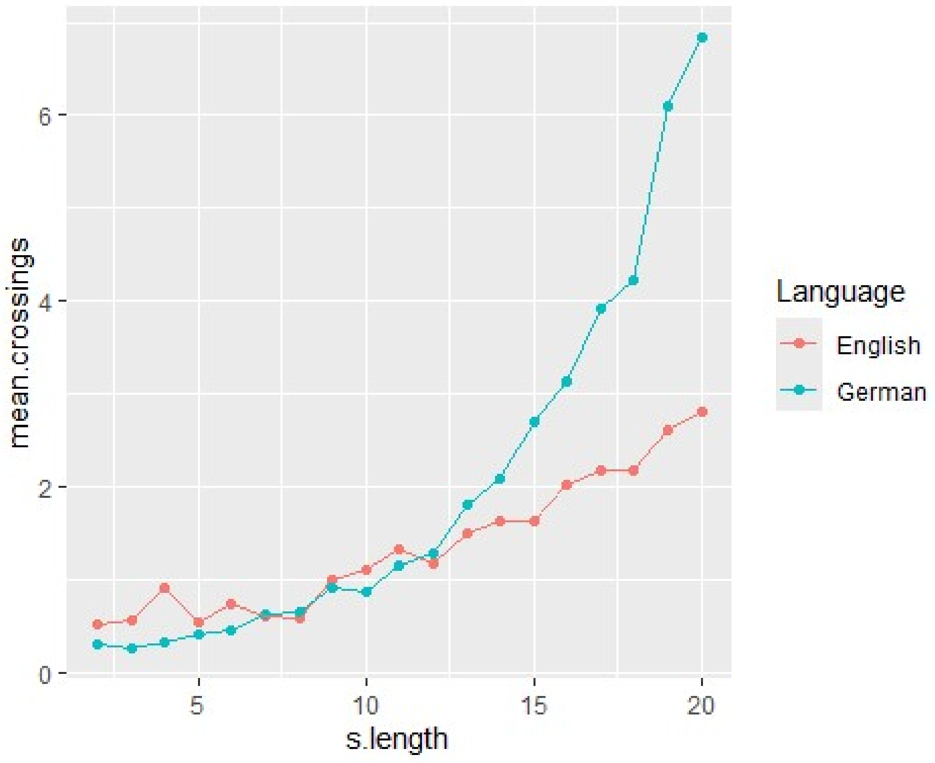


Exercise 2.4: Quantify evidence for the models M1 and M2 using k-fold cross-validation.

Using the sample code given in the problem for quantifying evidence for the models.

```
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

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


# 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.
```



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

```r
# These two vectors will store log predictive desnsities
# 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)
  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 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,]$s.length+post.m2[,3]*ytest[i,]$l
ang+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)
}

## 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.

# Predictive accuracy of model M1
elpd.m1 <- sum(lpds.m1)
elpd.m1

## [1] -2817.517

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

## [1] -2683.562

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

## [1] 133.9556
```

Seeing that the Evidence in favor for the model 2 over model 1 is positive, it implies that the model 2 out-performs model 1