Assignment 3 - Solution

Nicole Cruz & Julia Haaf

Preparation

```r
# packages
library("brms")
library("ggplot2")
library("tidyverse")

# data
load("data/income.RData")
```

data: `income.RData`

```r
head(income)
```

```
##    ls emotst inc_hh emotst_cat
## 1 3.8    6.0      7       high
## 2 1.0    3.0      6        low
## 3 5.0    4.0     10        low
## 4 4.2    3.5      7        low
## 5 4.4    6.0      2       high
## 6 3.8    6.5      7       high
```

The data set contains four variables

Variable	Description
`ls`	Life satisfaction (1-5)
`emotst`	Emotional stability (1-7)
`emotst_cat`	Emot. stability (high-low)
`inc_hh`	Household income (in 10.000)

Excercise 1

Fit the following Bayesian model to assess the effect of household income on life satisfaction.

```r
# Casewise exclusion of missing data
income <- na.exclude(income)

# This function can be used to figure out the structure of the priors for the sepcific model including
# default_prior(ls ~ inc_hh,
#               data = income)

bprior <- c(brms::prior(normal(3, 2), class = Intercept)
            , brms::prior(normal(0, 1), class = b, coef = inc_hh)
            , brms::prior(normal(0, 2.5), class = sigma))
```

```r
model.1 <- brm(ls ~ 1 + inc_hh
               , data = income
               , prior = bprior
               , silent = 2
               , refresh = 0)
```

```
## Trying to compile a simple C file

## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## using C compiler: ‘gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0’
## gcc -I"/usr/share/R/include" -DNDEBUG   -I"/home/juliahaaf/R/x86_64-pc-linux-gnu-library/4.4/Rcpp/in
## In file included from /home/juliahaaf/R/x86_64-pc-linux-gnu-library/4.4/RcppEigen/include/Eigen/Core
##                  from /home/juliahaaf/R/x86_64-pc-linux-gnu-library/4.4/RcppEigen/include/Eigen/Dens
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp:22,
##                  from <command-line>:
## /home/juliahaaf/R/x86_64-pc-linux-gnu-library/4.4/RcppEigen/include/Eigen/src/Core/util/Macros.h:679
##   679 | #include <cmath>
##       |          ^~~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:195: foo.o] Error 1
```

a) Consider the priors. Do you think they are a good choice? What would you change (if anything).

b) Ensure convergence and visuaalize the posterior distributions of all parameters. Interpret the results.

Solution

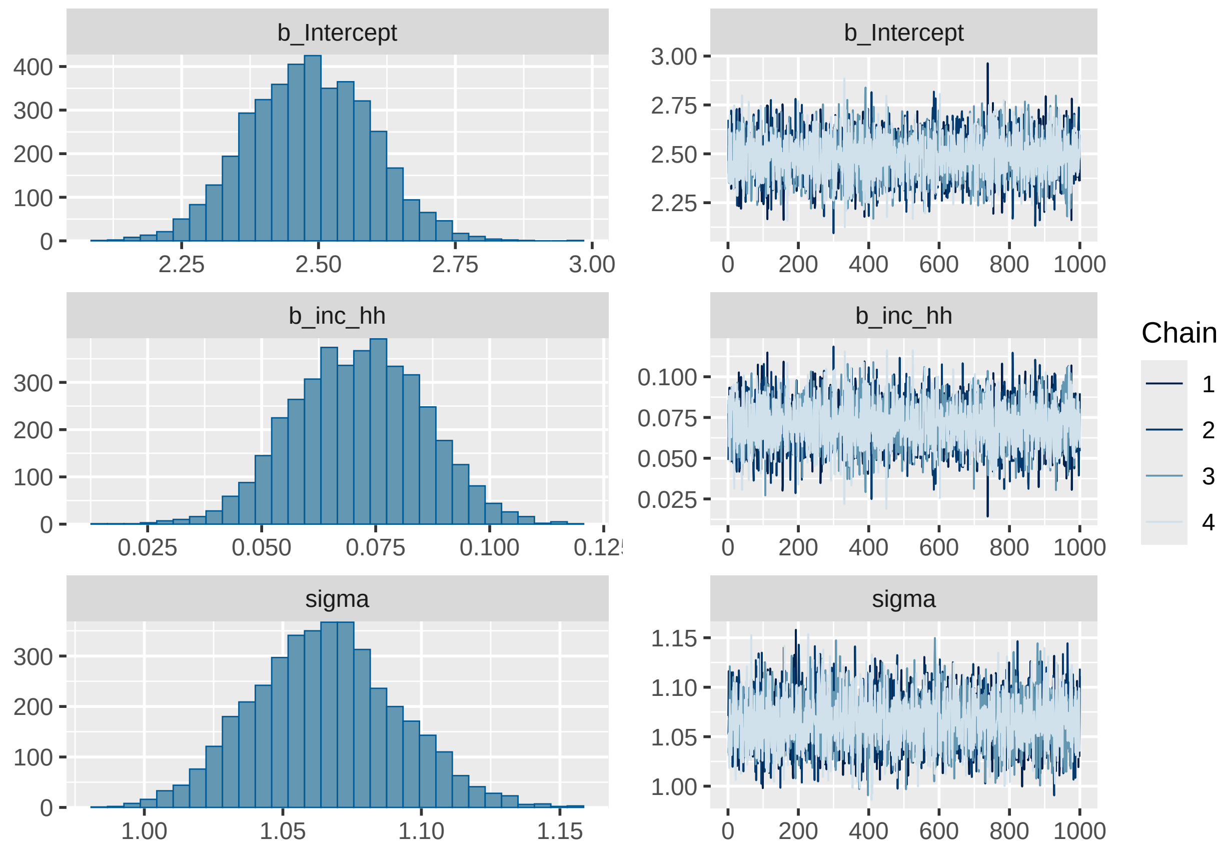
- Y_i : Life satisfaction of the ith person
- x_i : Income

$$Y_i \sim \text{Normal}(\mu_i, \sigma^2)$$
$$\mu_i = \beta_0 + x_i \beta_1$$

1a. The `ls`-values are between 1 and 5, so effects cannot be larger than one step on the life satisfaction scale. The mean is probably around 3. Sigma, the noise standard deviation can also not be bigger than 2-3 because of the range restriction of the data.

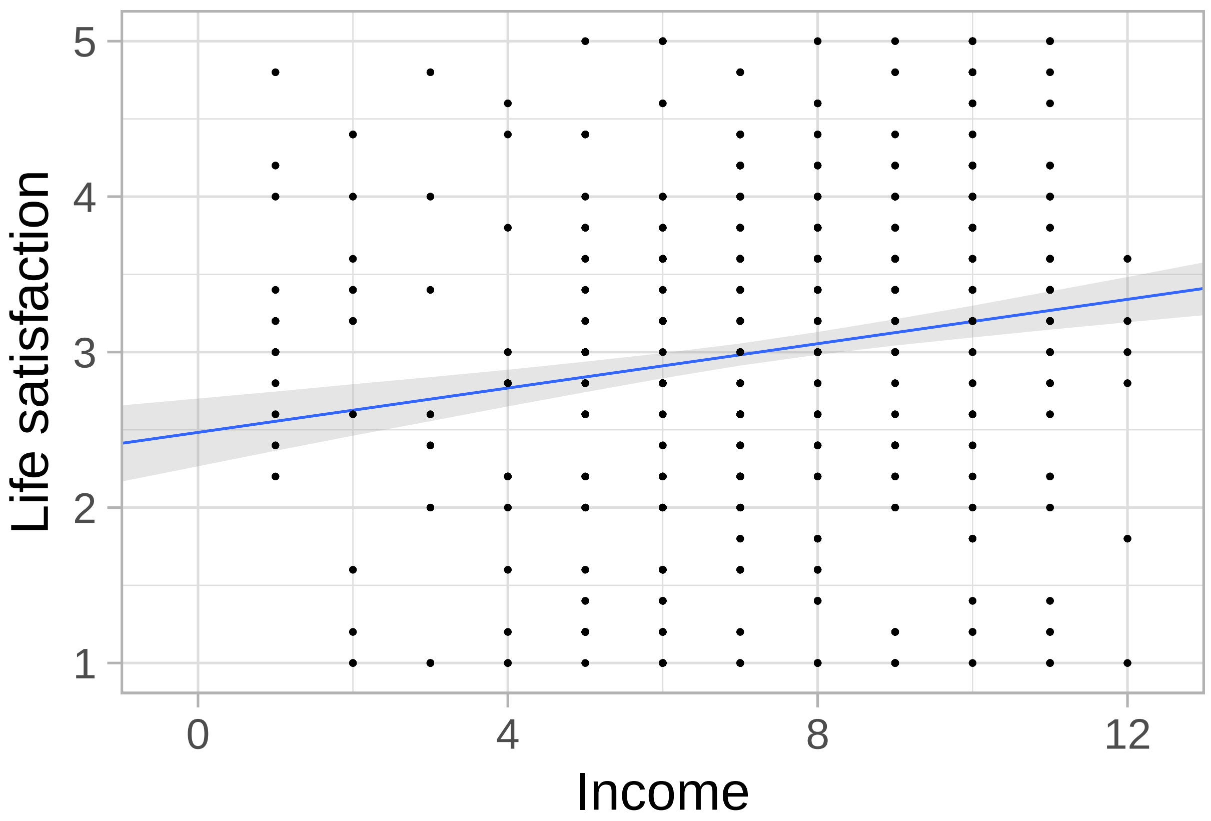
1b.

```r
plot(model.1)
```



None of the posterior credible intervals includes 0. For an increase of 10.000 Euros of household income there is an increase of life satsifaction of about 0.07 points.

```r
## Neue Daten werden erstellt, um die Regressionsgerade zu zeichnen
nd <- tibble(inc_hh = -1:13, 2)
## Regression wird auf die neuen Daten angewendet
f <- fitted(model.1, newdata = nd) %>%
  as_tibble() %>%
  bind_cols(nd)
income %>%
  ggplot(aes(x = inc_hh)) +
  geom_smooth(data = f,
              aes(y = Estimate, ymin = Q2.5, ymax = Q97.5),
              stat = "identity",
              alpha = 1/4, linewidth = 1/2) +
  geom_point(aes(y = ls),
             size = 2/3) +
  scale_x_continuous("Income", expand = c(0, 0)) +
  ylab("Life satisfaction") +
  theme_light(base_size = 20, )
```



c) Is there evidence that household income influences life satisfaction? Compare Model 1 to an alternative model without the income effect using Bayes factor.

d) Also compare the posterior prediction of the models. Is one of them better than the other?

Solution 1c.

```
bprior <- c(brms::prior(normal(3, 2), class = Intercept)
            , brms::prior(normal(0, 2.5), class = sigma))

model.0 <- brm(ls ~ 1
               , data = income
               , prior = bprior
               , silent = 2)
```

```
## Trying to compile a simple C file
```

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## using C compiler: 'gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0'
## gcc -I"/usr/share/R/include" -DNDEBUG   -I"/home/juliahaaf/R/x86_64-pc-linux-gnu-library/4.4/Rcpp/in
## In file included from /home/juliahaaf/R/x86_64-pc-linux-gnu-library/4.4/RcppEigen/include/Eigen/Core
##                  from /home/juliahaaf/R/x86_64-pc-linux-gnu-library/4.4/RcppEigen/include/Eigen/Dens
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp:22,
##                  from <command-line>:
## /home/juliahaaf/R/x86_64-pc-linux-gnu-library/4.4/RcppEigen/include/Eigen/src/Core/util/Macros.h:679
##   679 | #include <cmath>
##       |          ^~~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:195: foo.o] Error 1
##
```

```
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000128 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.28 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)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.108 seconds (Warm-up)
## Chain 1:                0.119 seconds (Sampling)
## Chain 1:                0.227 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2.8e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.28 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.103 seconds (Warm-up)
## Chain 2:                0.088 seconds (Sampling)
## Chain 2:                0.191 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 3e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.3 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.103 seconds (Warm-up)
## Chain 3:                0.107 seconds (Sampling)
## Chain 3:                0.21 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 3e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.3 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.102 seconds (Warm-up)
## Chain 4:                0.086 seconds (Sampling)
## Chain 4:                0.188 seconds (Total)
## Chain 4:
```

```
bayes_factor(model.1, model.0)
```

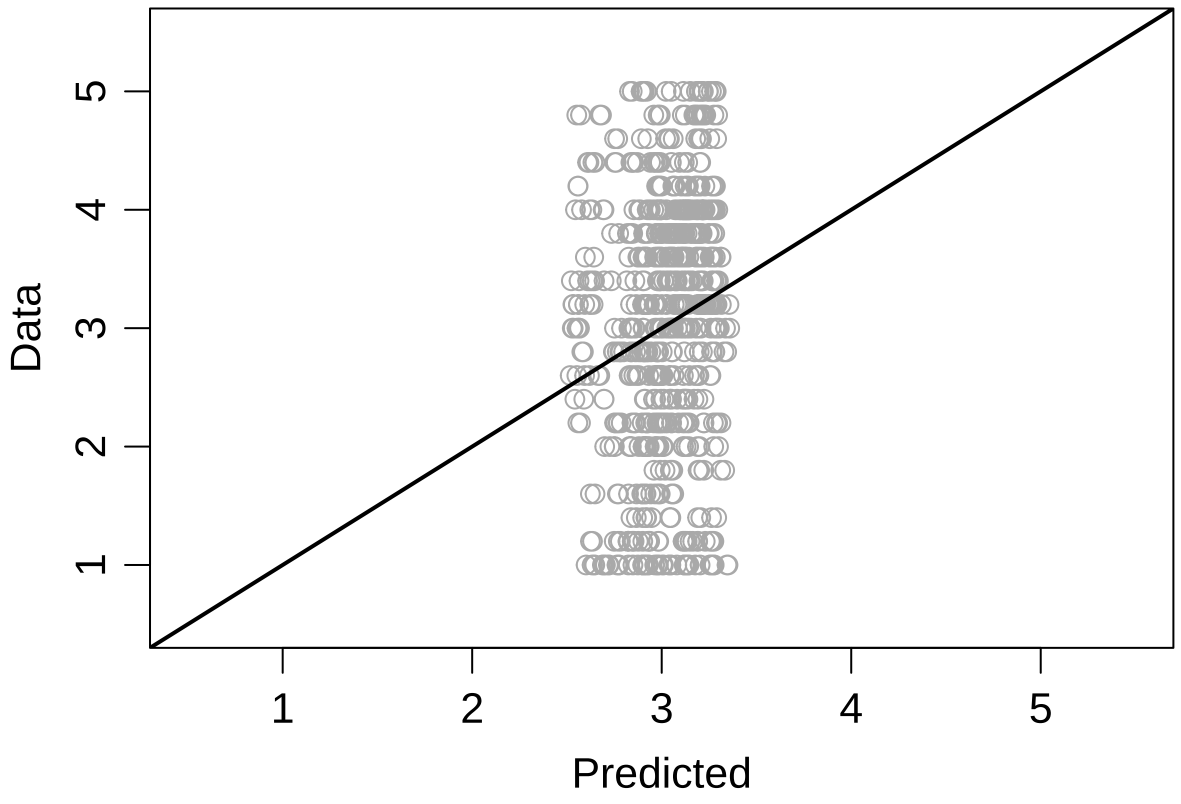
```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
```

```
## Estimated Bayes factor in favor of model.1 over model.0: 3276.53737
```

1d.

```
post.pred.1 <- posterior_predict(model.1)
post.pred.0 <- posterior_predict(model.0)
```

```
par(mar = c(4,4,.5,.5), cex = 1.3, mgp = c(2, .7, 0))
post.pred.mean.1 <- colMeans(post.pred.1)
plot(post.pred.mean.1
     , income$ls
     , ylab = "Data", xlab = "Predicted"
     , pch = 1, col = "darkgray"
     , ylim = c(.5, 5.5), xlim = c(.5, 5.5))
abline(0, 1, lwd = 2)
```

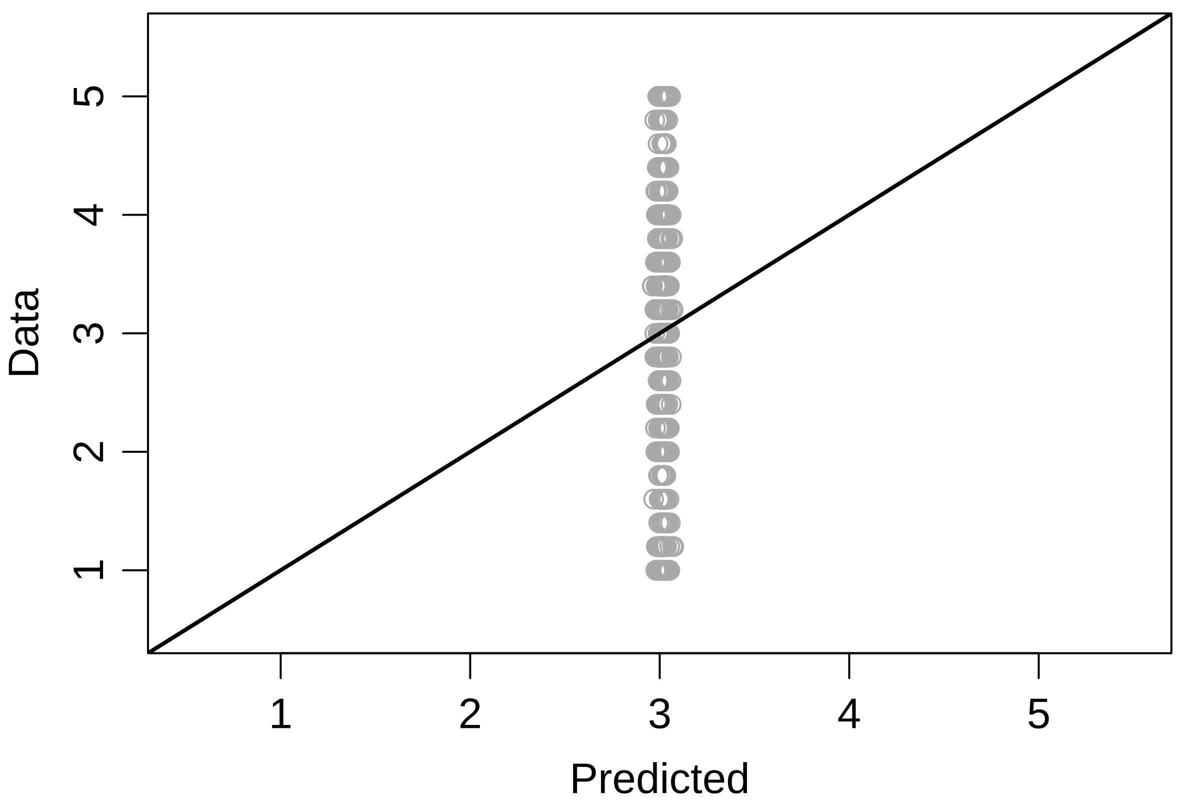


```
cor(post.pred.mean.1, income$ls)
```

```
## [1] 0.1700828
```

```
par(mar = c(4,4,.5,.5), cex = 1.3, mgp = c(2, .7, 0))
post.pred.mean.0 <- colMeans(post.pred.0)
plot(post.pred.mean.0
     , income$ls
     , ylab = "Data", xlab = "Predicted"
     , pch = 1, col = "darkgray"
     , ylim = c(.5, 5.5), xlim = c(.5, 5.5))
```

```r
abline(0, 1, lwd = 2)
```



```r
cor(post.pred.mean.0, income$ls)
```

```
## [1] 0.03493745
```