5.6 evaluating, checking, comparing

Chris Parrish

July 3, 2016

Contents

5.6 evaluating, checking, comparing

reference:
- ARM chapter 05, github

```
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
library(ggplot2)
```

residuals

$$residual_i = y_i - E(y_i \mid X_i) = y_i - logit^{-1}(X_i\beta)$$

evaluating, checking, comparing

data

```
### Data
source("wells.data.R", echo = TRUE)
```

```
##
## > N <- 3020
##
## > switched <- c(1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## +     1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## +     1, 1, 0, 0, 1, 0, 1, 1, 0,  .... [TRUNCATED]
##
## > arsenic <- c(2.36, 0.71, 2.07, 1.15, 1.1, 3.9, 2.97,
## +     3.24, 3.28, 2.52, 3.13, 3.04, 2.91, 3.21, 1.7, 1.8, 1.44,
## +     1.43, 2.33, 2.83, 1.79, .... [TRUNCATED]
##
## > dist <- c(16.826000213623, 47.3219985961914, 20.9669990539551,
## +     21.4860000610352, 40.8740005493164, 69.5179977416992, 80.7109985351563,
## +     .... [TRUNCATED]
##
## > assoc <- c(0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1,
## +     1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0,
## +     1, 0, 0, 1, 1, 0, 1, 0, 0,  .... [TRUNCATED]
##
## > educ <- c(0, 0, 10, 12, 14, 9, 4, 10, 0, 0, 5, 0,
## +     0, 0, 0, 7, 7, 7, 0, 10, 7, 0, 5, 0, 8, 8, 10, 16, 10, 10,
## +     10, 10, 0, 0, 0, 3, 0, 10 .... [TRUNCATED]
```

model

wells_predicted.stan

```
data {
  int<lower=0> N;
  int<lower=0,upper=1> switched[N];
  vector[N] dist;
  vector[N] arsenic;
  vector[N] educ;
}
transformed data {
  vector[N] c_dist100;
  vector[N] c_arsenic;
  vector[N] c_educ4;
  vector[N] da_inter;
  vector[N] de_inter;
  vector[N] ae_inter;
  c_dist100 = (dist - mean(dist)) / 100.0;
  c_arsenic = arsenic - mean(arsenic);        // no log transformation
  c_educ4   = (educ - mean(educ)) / 4.0;
  da_inter  = c_dist100 .* c_arsenic;
  de_inter  = c_dist100 .* c_educ4;
  ae_inter  = c_arsenic .* c_educ4;
}
parameters {
  vector[7] beta;
}
model {
  switched ~ bernoulli_logit(beta[1] +
                             beta[2] * c_dist100 +
                             beta[3] * c_arsenic +
                             beta[4] * c_educ4 +
                             beta[5] * da_inter +
                             beta[6] * de_inter +
                             beta[7] * ae_inter);
}
generated quantities {
  vector[N] pred;
  for (i in 1:N)
    pred[i] = inv_logit(beta[1] +
                        beta[2] * c_dist100[i] +
                        beta[3] * c_arsenic[i] +
                        beta[4] * c_educ4[i] +
                        beta[5] * da_inter[i] +
                        beta[6] * de_inter[i] +
                        beta[7] * ae_inter[i]);
}
```

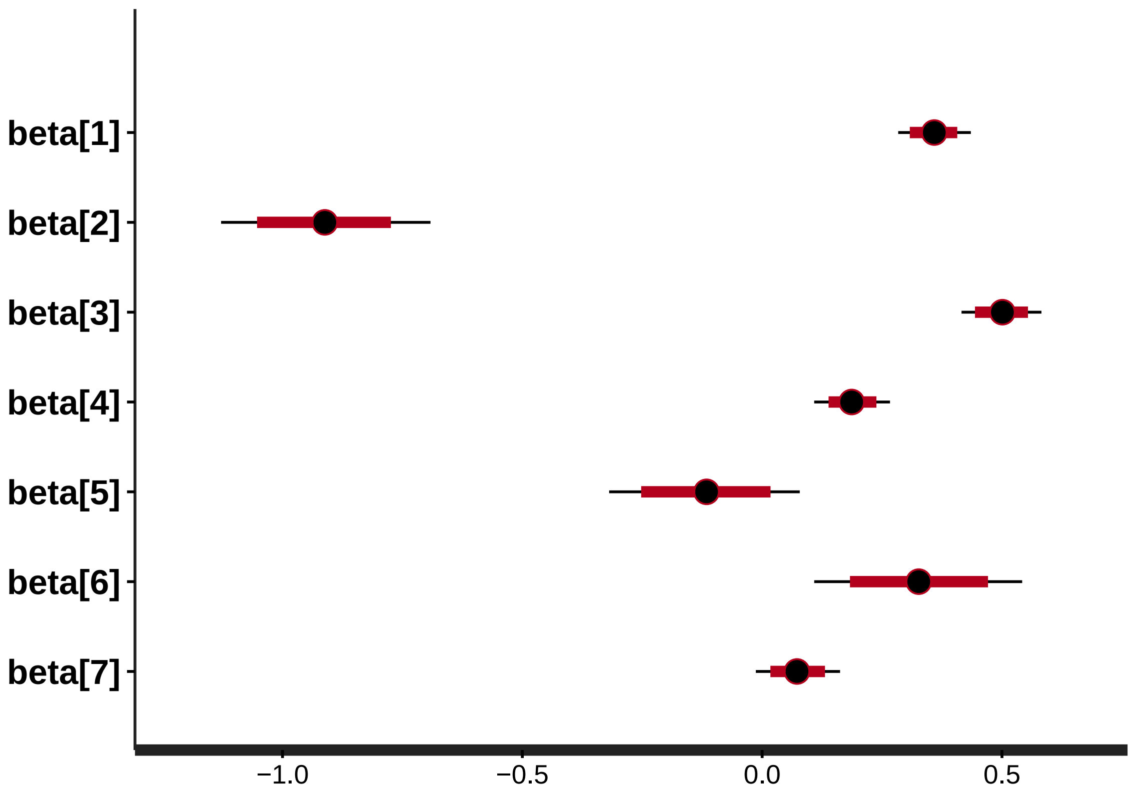
fit

```
### Model: switched ~ c_dist100 + c_arsenic + c_educ4 + c_dist100:c_arsenic
###                      + c_dist100:c_educ4 + c_arsenic:c_educ4
### c_dist100 <- (dist - mean(dist)) / 100
### c_arsenic <- arsenic - mean(arsenic)
### c_educ4   <- (educ - mean(educ)) / 4
data.list <- c("N", "switched", "dist", "arsenic", "educ")
wells_predicted.sf <- stan(file='wells_predicted.stan', data=data.list,
                           iter=1000, chains=4)
```

```
plot(wells_predicted.sf, pars = "beta")
```

```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```



```r
# pairs(wells_predicted.sf)
print(wells_predicted.sf, pars = c("beta", "lp__"))
```

```
## Inference for Stan model: wells_predicted.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##              mean se_mean   sd     2.5%      25%      50%      75%    97.5%
## beta[1]      0.36    0.00 0.04     0.28     0.33     0.36     0.38     0.44
## beta[2]     -0.91    0.00 0.11    -1.13    -0.98    -0.91    -0.84    -0.69
## beta[3]      0.50    0.00 0.04     0.42     0.47     0.50     0.53     0.58
## beta[4]      0.19    0.00 0.04     0.11     0.16     0.19     0.21     0.27
## beta[5]     -0.12    0.00 0.10    -0.32    -0.19    -0.12    -0.05     0.08
## beta[6]      0.33    0.00 0.11     0.11     0.25     0.33     0.40     0.54
## beta[7]      0.07    0.00 0.05    -0.01     0.04     0.07     0.11     0.16
## lp__     -1949.47    0.06 1.90 -1954.02 -1950.53 -1949.13 -1948.07 -1946.82
##          n_eff Rhat
## beta[1]   1678    1
## beta[2]   2000    1
## beta[3]   1901    1
## beta[4]   1699    1
## beta[5]   1762    1
## beta[6]   1842    1
## beta[7]   2000    1
## lp__      1105    1
##
## Samples were drawn using NUTS(diag_e) at Tue Jul  5 23:08:51 2016.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
##  The estimated Bayesian Fraction of Missing Information is a measure of
##  the efficiency of the sampler with values close to 1 being ideal.
##  For each chain, these estimates are
##  1 1 1 1
```

figure 5.13a

```
## Residual Plot (Figure 5.13 (a))
prob.pred.1 <- colMeans(extract(wells_predicted.sf, "pred")$pred)
wells_resid.ggdf.1 <- data.frame(prob = prob.pred.1,
                                 resid = switched - prob.pred.1)
p1 <- ggplot(wells_resid.ggdf.1, aes(prob, resid)) +
    geom_point(shape = 20, color = "darkred") +
    scale_x_continuous("Estimated Pr(switching)", limits = c(0, 1),
                       breaks = seq(0, 1, 0.2)) +
    scale_y_continuous("Observed - estimated", limits = c(-1, 1),
                       breaks = seq(-1, 1, 0.5)) +
    ggtitle("Residual plot")
print(p1)
```

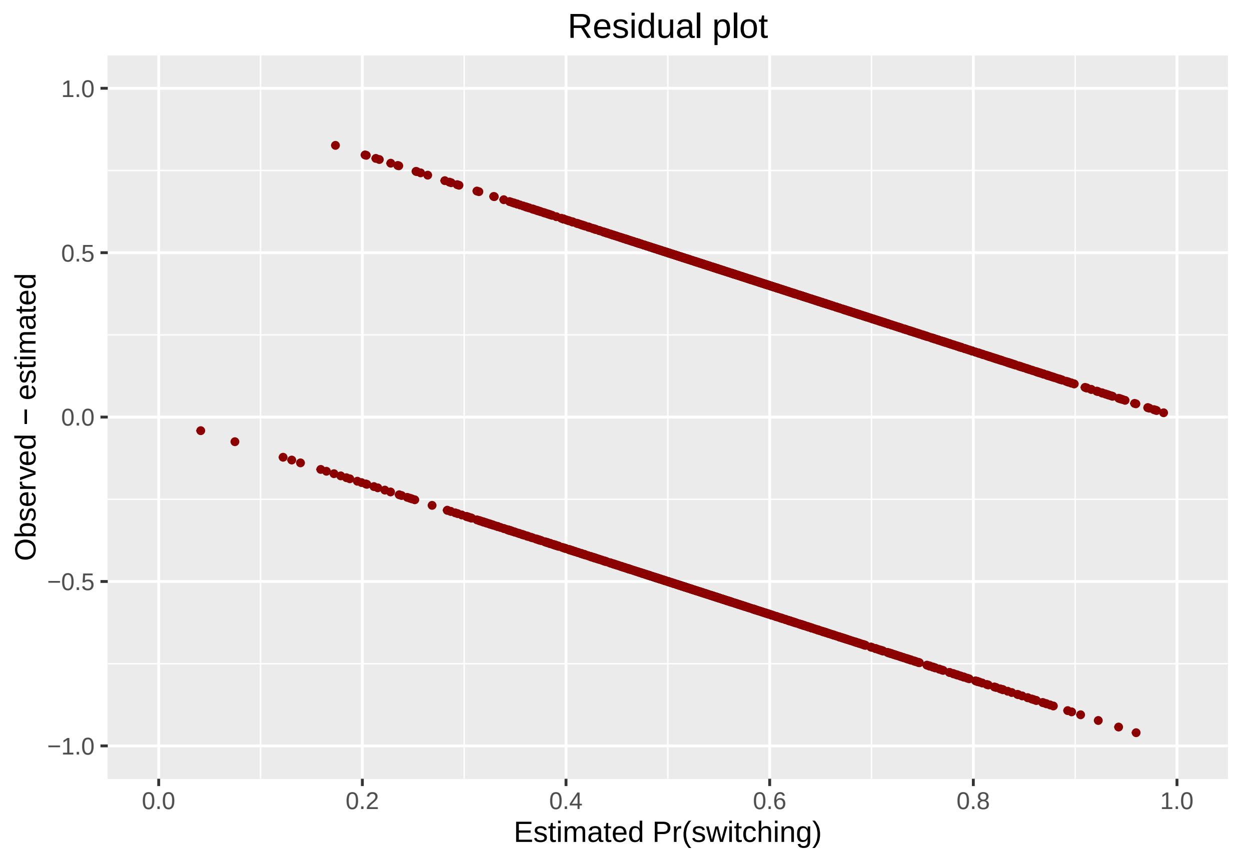


figure 5.13b

```r
## Binned residual plot
# Defining binned residuals
binned.resids <- function (x, y, nclass = sqrt(length(x))) {
    breaks.index <- floor(length(x) * (1:(nclass-1)) / nclass)
    breaks <- c (-Inf, sort(x)[breaks.index], Inf)
    output <- NULL
    xbreaks <- NULL
    x.binned <- as.numeric(cut (x, breaks))
    for (i in 1:nclass) {
        items <- (1:length(x))[x.binned == i]
        x.range <- range(x[items])
        xbar <- mean(x[items])
        ybar <- mean(y[items])
        n <- length(items)
        sdev <- sd(y[items])
        output <- rbind(output, c(xbar, ybar, n, x.range, 2 * sdev / sqrt(n)))
    }
    colnames (output) <- c("xbar", "ybar", "n", "x.lo", "x.hi", "2se")
    return(list(binned = output, xbreaks = xbreaks))
}
```

```r
# Binned residuals vs. estimated probability of switched (Figure 5.13 (b))
# dev.new()
br <- binned.resids(prob.pred.1, switched - prob.pred.1, nclass = 40)$binned
binned.ggdf.1 <- data.frame(x = br[,1], y = br[,2], disp = br[,6])
p2 <- ggplot(binned.ggdf.1, aes(x, y)) +
    geom_point(shape = 20, color = "darkred") +
    geom_line(aes(x = x, y = disp), color = "gray") +
    geom_line(aes(x = x, y = - disp), color = "gray") +
    geom_hline(yintercept = 0, color = "gray") +
    scale_x_continuous("Estimated Pr(switching)", breaks = seq(0.3, 0.9, 0.1)) +
    scale_y_continuous("Average residual") +
    ggtitle("Binned residual plot")
print(p2)
```

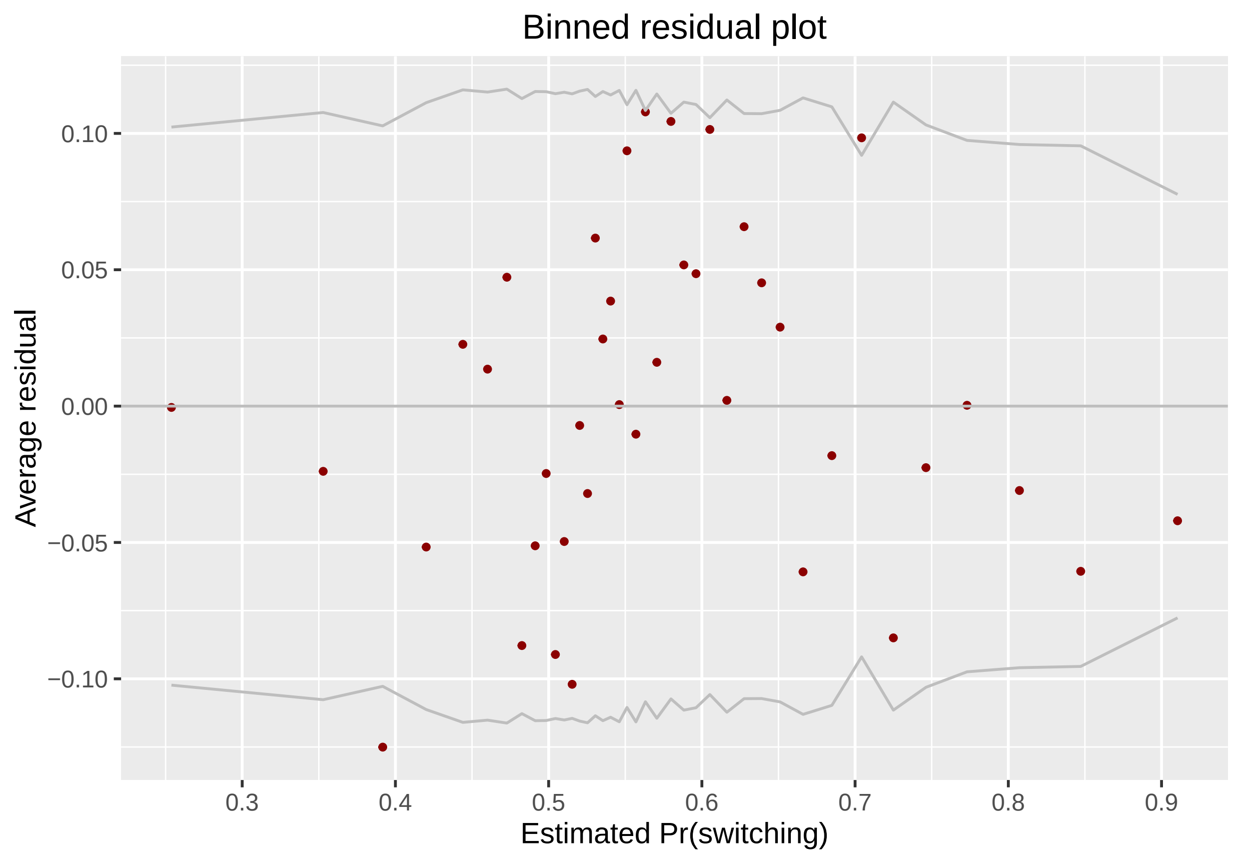


figure 5.14a

```
## Plot of binned residuals vs. inputs of interest
# distance (Figure 5.13 (a))
# dev.new()
br.dist <- binned.resids(dist, switched - prob.pred.1, nclass = 40)$binned
binned.ggdf.2 <- data.frame(x = br.dist[,1], y = br.dist[,2], disp = br.dist[,6])
p3 <- ggplot(binned.ggdf.2, aes(x, y)) +
    geom_point(shape = 20, color = "darkred") +
    geom_line(aes(x = x, y = disp), color = "gray") +
    geom_line(aes(x = x, y = - disp), color = "gray") +
    geom_hline(yintercept = 0, color = "gray") +
    scale_x_continuous("Distance to nearest safe well", breaks = seq(0, 150, 50)) +
    scale_y_continuous("Average residual") +
    ggtitle("Binned residual plot")
print(p3)
```

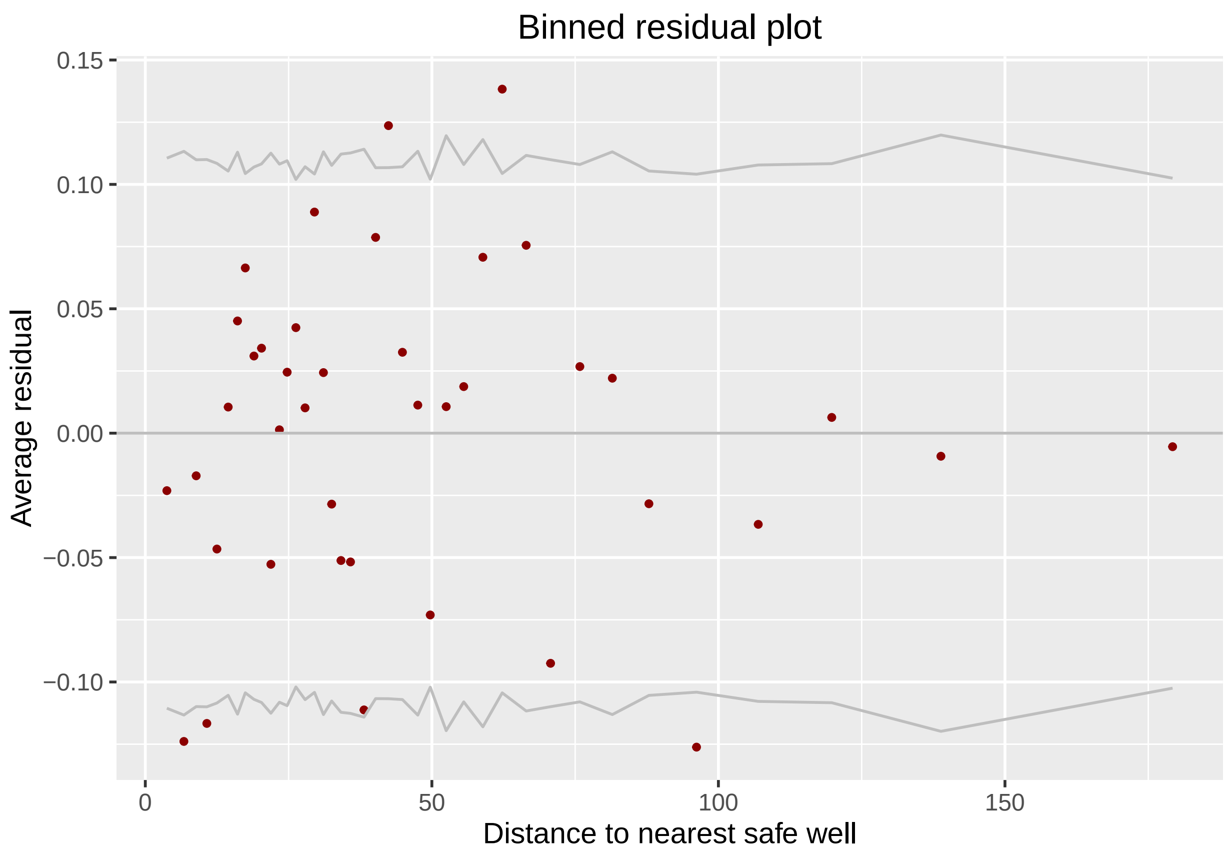
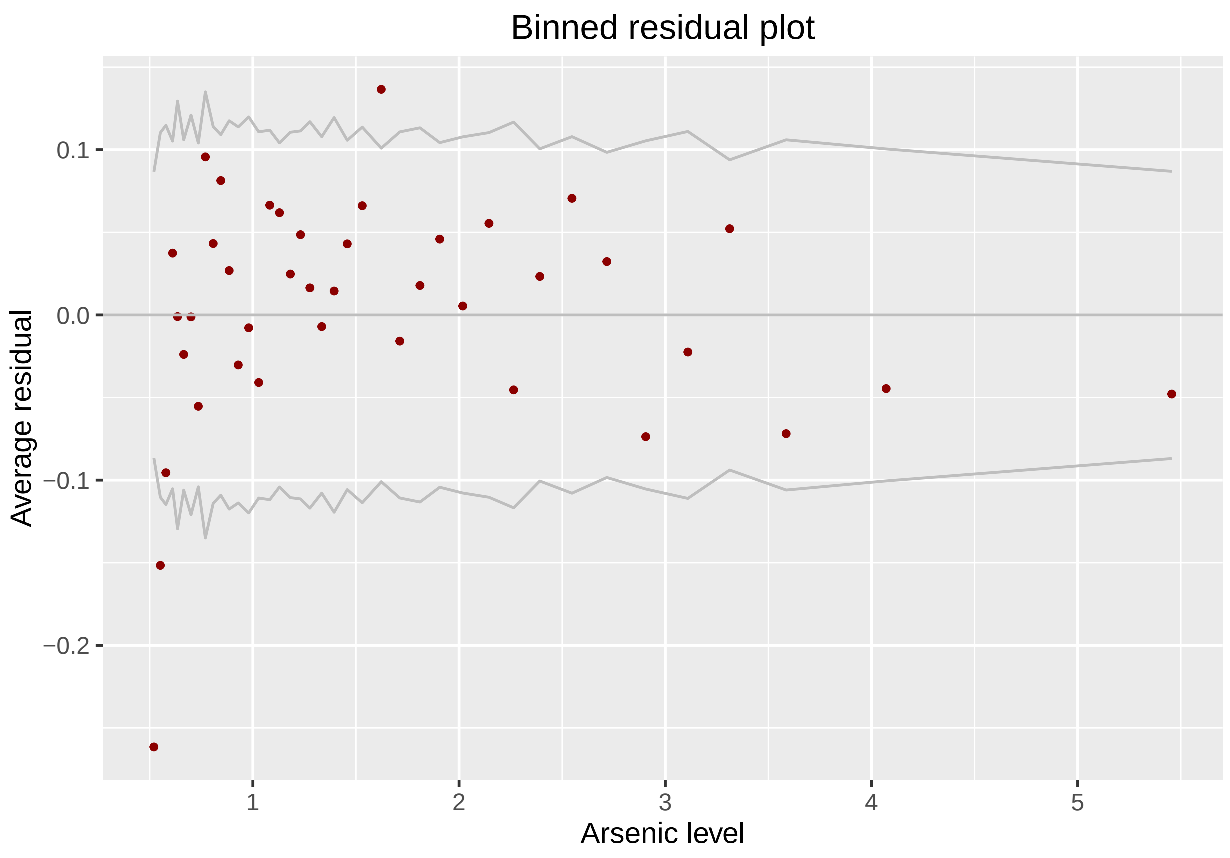


figure 5.14b

```
# arsenic (Figure 5.13 (b))
# dev.new()
br.as <- binned.resids(arsenic, switched - prob.pred.1, nclass = 40)$binned
binned.ggdf.3 <- data.frame(x = br.as[,1], y = br.as[,2], disp = br.as[,6])
p4 <- ggplot(binned.ggdf.3, aes(x, y)) +
    geom_point(shape = 20, color = "darkred") +
    geom_line(aes(x = x, y = disp), color = "gray") +
    geom_line(aes(x = x, y = - disp), color = "gray") +
    geom_hline(yintercept = 0, color = "gray") +
    scale_x_continuous("Arsenic level", breaks = seq(0, 5)) +
    scale_y_continuous("Average residual") +
    ggtitle("Binned residual plot")
print(p4)
```



log transformation

model

wells_predicted_log.stan

```
data {
  int<lower=0> N;
  int<lower=0,upper=1> switched[N];
  vector[N] dist;
  vector[N] arsenic;
  vector[N] educ;
}
transformed data {
  vector[N] c_dist100;
  vector[N] log_arsenic;
  vector[N] c_log_arsenic;
  vector[N] c_educ4;
  vector[N] da_inter;
  vector[N] de_inter;
  vector[N] ae_inter;
  c_dist100     = (dist - mean(dist)) / 100.0;
  log_arsenic   = log(arsenic);
  c_log_arsenic = log_arsenic - mean(log_arsenic);
  c_educ4       = (educ - mean(educ)) / 4.0;
  da_inter      = c_dist100 .* c_log_arsenic;
  de_inter      = c_dist100 .* c_educ4;
  ae_inter      = c_log_arsenic .* c_educ4;
}
parameters {
  vector[7] beta;
}
model {
  switched ~ bernoulli_logit(beta[1] +
                             beta[2] * c_dist100 +
                             beta[3] * c_log_arsenic +
                             beta[4] * c_educ4 +
                             beta[5] * da_inter +
                             beta[6] * de_inter +
                             beta[7] * ae_inter);
}
generated quantities {
  vector[N] pred;
  for (i in 1:N)
    pred[i] = inv_logit(beta[1] +
                        beta[2] * c_dist100[i] +
                        beta[3] * c_log_arsenic[i] +
                        beta[4] * c_educ4[i] +
                        beta[5] * da_inter[i] +
                        beta[6] * de_inter[i] +
                        beta[7] * ae_inter[i]);
}
```

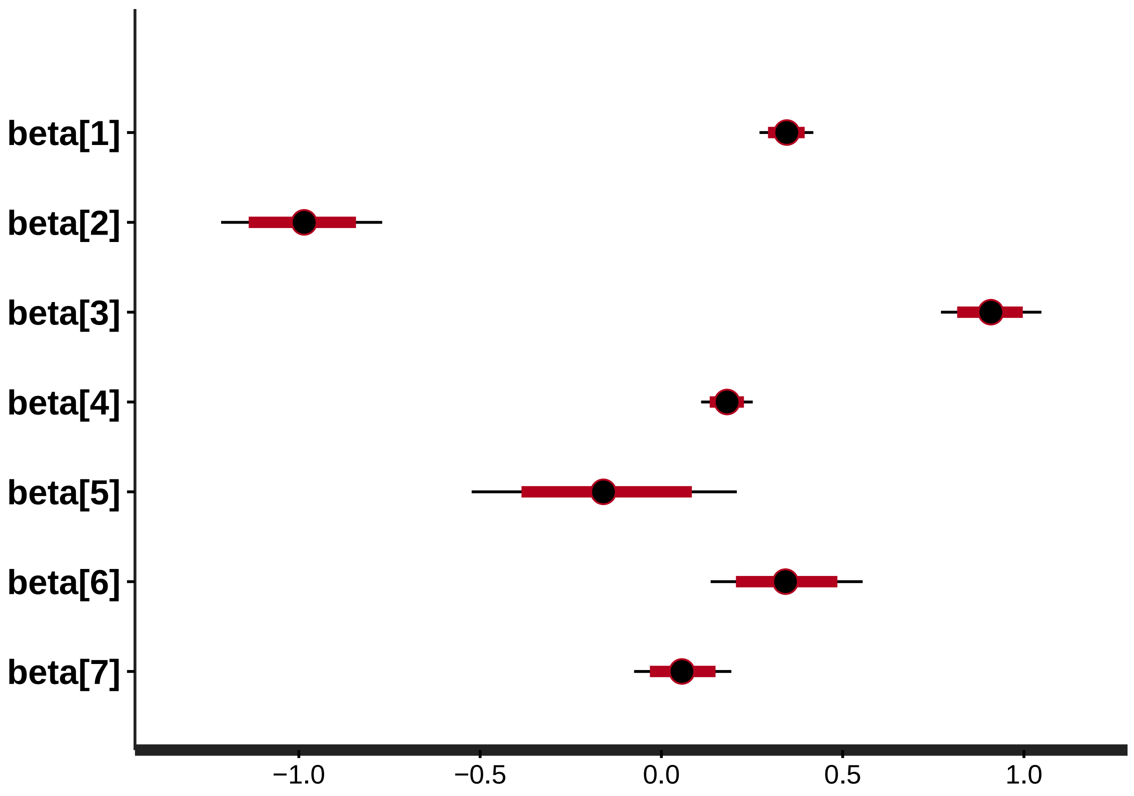
fit

```
### Log transformation: switched ~ c_dist100 + c_log_arsenic + c_educ4
###                                 + c_dist100:c_log_arsenic + c_dist100:c_educ4
###                                 + c_log_arsenic:c_educ4
### c_log_arsenic <- log(arsenic) - mean(log(arsenic))
wells_predicted_log.sf <- stan(file='wells_predicted_log.stan',
                               data=data.list,
                               iter=1000, chains=4)
```

```
plot(wells_predicted_log.sf, pars = "beta")
```

```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```



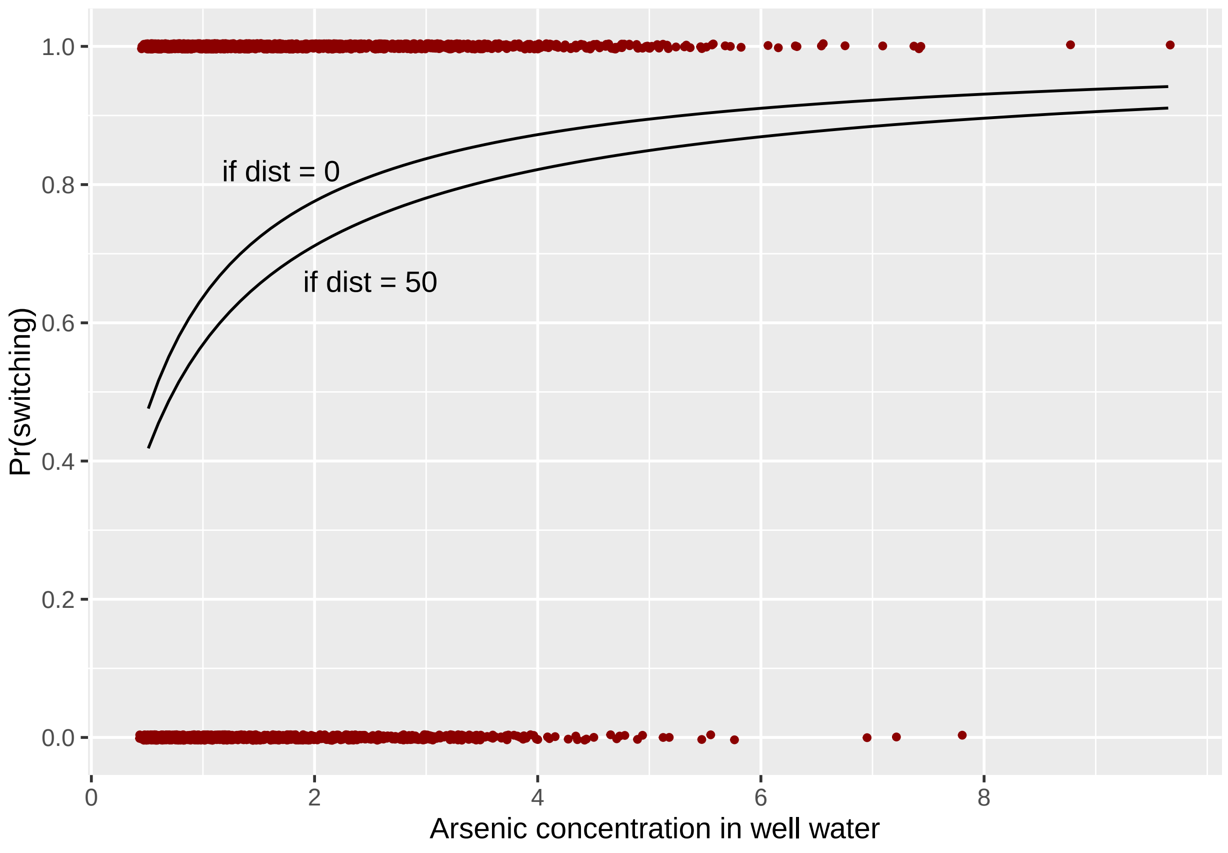
```r
# pairs(wells_predicted_log.sf)
print(wells_predicted_log.sf, pars = c("beta", "lp__"))
```

```
## Inference for Stan model: wells_predicted_log.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##               mean se_mean   sd     2.5%      25%      50%      75%    97.5%
## beta[1]       0.34    0.00 0.04     0.27     0.32     0.35     0.37     0.42
## beta[2]      -0.99    0.00 0.11    -1.21    -1.07    -0.99    -0.91    -0.77
## beta[3]       0.91    0.00 0.07     0.77     0.86     0.91     0.96     1.05
## beta[4]       0.18    0.00 0.04     0.11     0.16     0.18     0.21     0.25
## beta[5]      -0.16    0.00 0.19    -0.52    -0.28    -0.16    -0.03     0.21
## beta[6]       0.35    0.00 0.11     0.14     0.27     0.34     0.42     0.56
## beta[7]       0.06    0.00 0.07    -0.08     0.01     0.06     0.11     0.19
## lp__      -1935.06    0.05 1.81 -1939.40 -1936.03 -1934.75 -1933.73 -1932.50
##         n_eff Rhat
## beta[1]  1674 1.00
## beta[2]  1460 1.00
## beta[3]  1598 1.00
## beta[4]  2000 1.00
## beta[5]  1752 1.00
## beta[6]  1909 1.00
## beta[7]  1862 1.01
## lp__     1116 1.00
##
## Samples were drawn using NUTS(diag_e) at Tue Jul  5 23:09:08 2016.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
##  The estimated Bayesian Fraction of Missing Information is a measure of
##  the efficiency of the sampler with values close to 1 being ideal.
##  For each chain, these estimates are
##  1 1 1 1.2
```

```r
beta.post <- extract(wells_predicted_log.sf, "beta")$beta
beta.mean <- colMeans(beta.post)
```

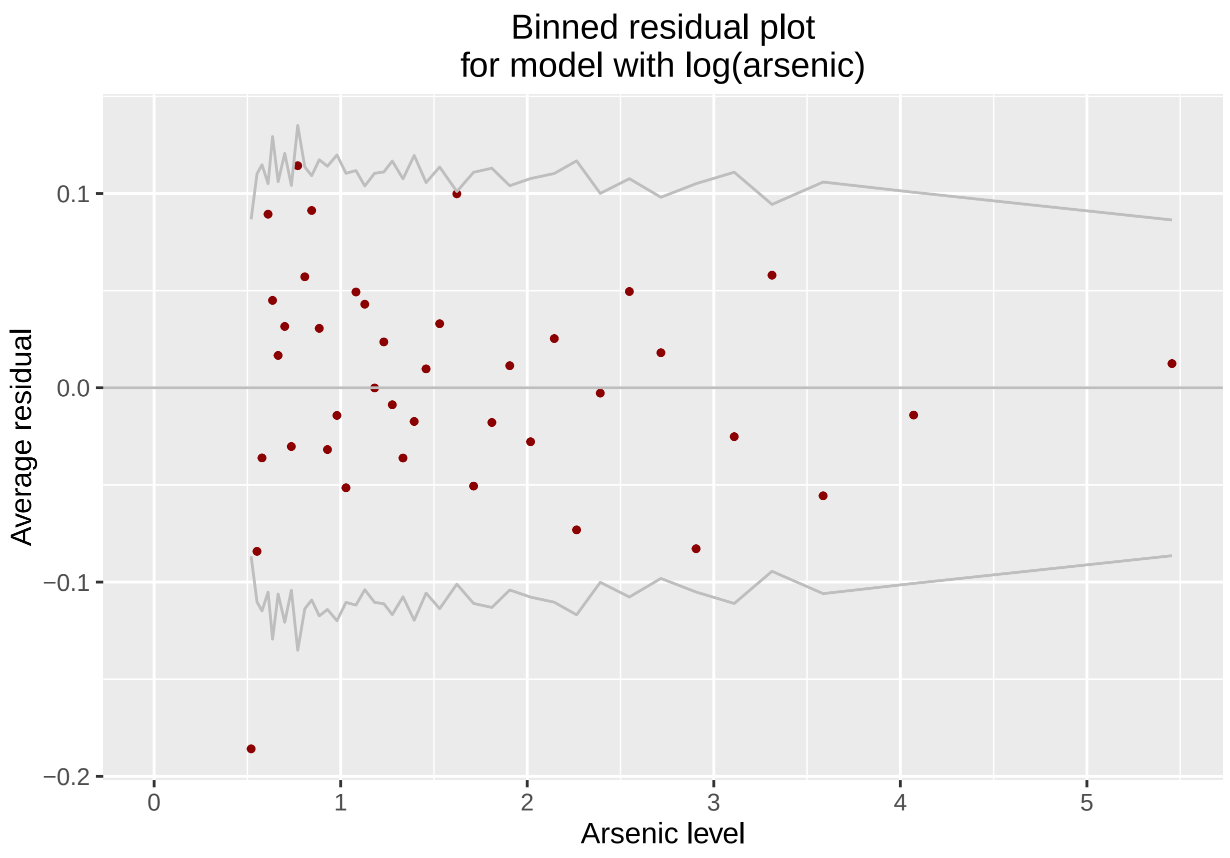
plot 5.15a

```r
## Graph for log model (Figure 5.15 (a))
# dev.new()
p5 <- ggplot(data.frame(switched, arsenic), aes(arsenic, switched)) +
    geom_jitter(position = position_jitter(width = 0.2, height = 0.01),
                shape = 20, color = "darkred") +
    stat_function(fun = function(x)
                  1 / (1 + exp(
                      - cbind(1, 0, log(x), mean(educ / 4), 0 * log(x),
                              0 * mean(educ / 4), log(x) * mean(educ / 4))
                      %*% beta.mean))) +
    stat_function(fun = function(x)
                  1 / (1 + exp(
                      - cbind(1, 0.5, log(x), mean(educ / 4), 0.5 * log(x),
                              0.5 * mean(educ / 4), log(x) * mean(educ / 4))
                      %*% beta.mean))) +
    annotate("text", x = c(1.7,2.5), y = c(0.82, 0.66),
             label = c("if dist = 0", "if dist = 50"), size = 4) +
    scale_x_continuous("Arsenic concentration in well water",
                       breaks = seq(from = 0, by = 2, length.out = 5)) +
    scale_y_continuous("Pr(switching)", breaks = seq(0, 1, 0.2))
print(p5)
```



plot 5.15b

```
## Graph of binned residuals for log model (Figure 5.15 (b))
# dev.new()
prob.pred.2 <- colMeans(extract(wells_predicted_log.sf, "pred")$pred)
br.log <- binned.resids(arsenic, switched - prob.pred.2, nclass = 40)$binned
binned.ggdf.2 <- data.frame(x = br.log[,1], y = br.log[,2], disp = br.log[,6])
p6 <- ggplot(binned.ggdf.2, aes(x, y)) +
    geom_point(shape = 20, color = "darkred") +
    geom_line(aes(x = x, y = disp), color = "gray") +
    geom_line(aes(x = x, y = - disp), color = "gray") +
    geom_hline(yintercept = 0, color = "gray") +
    scale_x_continuous("Arsenic level", limits = c(0, max(br.log[,1])),
                       breaks = seq(0, 5)) +
    scale_y_continuous("Average residual") +
    ggtitle("Binned residual plot\nfor model with log(arsenic)")
print(p6)
```



error rate

```
### Error rate
error.rate <- mean((prob.pred.2 > 0.5 & switched == 0) |
                   (prob.pred.2 < 0.5 & switched == 1))
error.rate
```

```
## [1] 0.3649007
```