7.4 predictive simulation for glm

Chris Parrish

July 3, 2016

Contents

7.4 predictive simulation for glm

reference:
- ARM chapter 07, github

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

predictive simulation for glm

data

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

```
##
## > "switc" <- 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, 1 .... [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, 1.7 .... [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,  .... [TRUNCATED]
##
## > "N" <- 3020
```

model

wells.stan

```
data {
  int<lower=0> N;
  vector[N] dist;
  int<lower=0,upper=1> switc[N];
}
parameters {
  vector[2] beta;
}
model {
  switc ~ bernoulli_logit(beta[1] + beta[2] * dist);   // distances in meters
}
```

fit

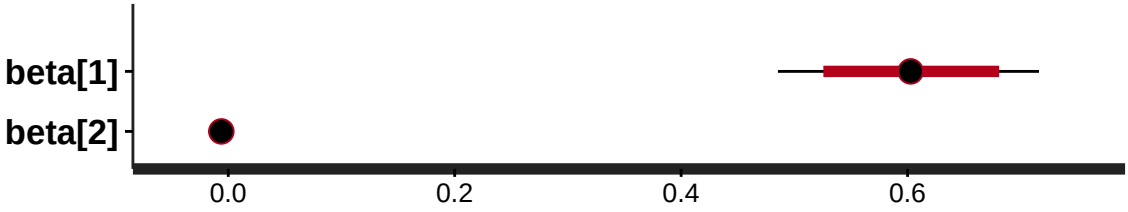
```
## Logistic regression (wells.stan)
## glm (switch ~ dist, family=binomial(link="logit"))
dataList.1 <- c("N","dist","switc")
```

```
wells.sf1 <- stan(file='wells.stan', data=dataList.1,
                  iter=1000, chains=4)
```

```
plot(wells.sf1)
```

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

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



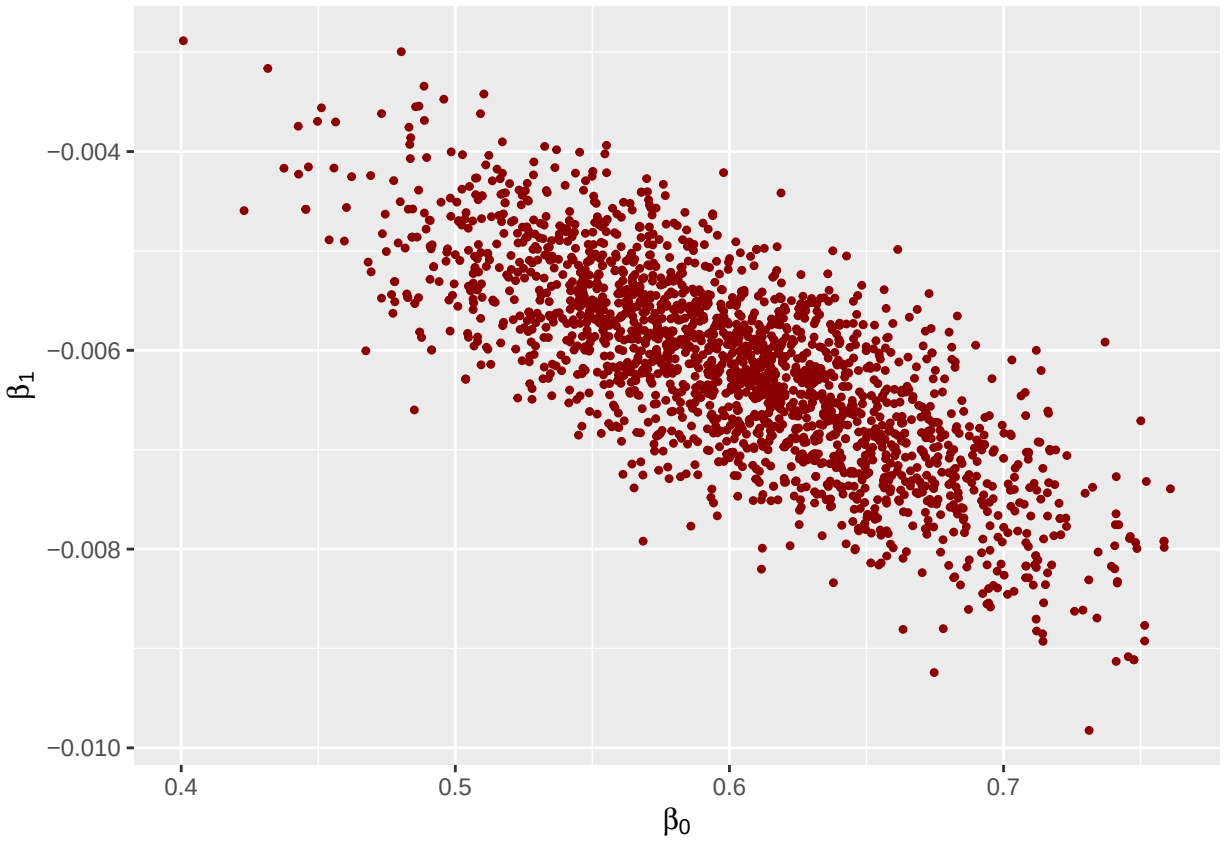

```
print(wells.sf1)
```

```
## Inference for Stan model: wells.
## 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]     0.60    0.00 0.06     0.49     0.56     0.60     0.64     0.72
## beta[2]    -0.01    0.00 0.00    -0.01    -0.01    -0.01    -0.01     0.00
## lp__    -2039.18    0.04 1.04 -2041.95 -2039.63 -2038.86 -2038.42 -2038.14
##         n_eff Rhat
## beta[1]   395 1.02
## beta[2]   651 1.01
## lp__      619 1.00
##
## Samples were drawn using NUTS(diag_e) at Thu Jul  7 13:22:27 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
##  0.9 1.1 0.8 0.9
```

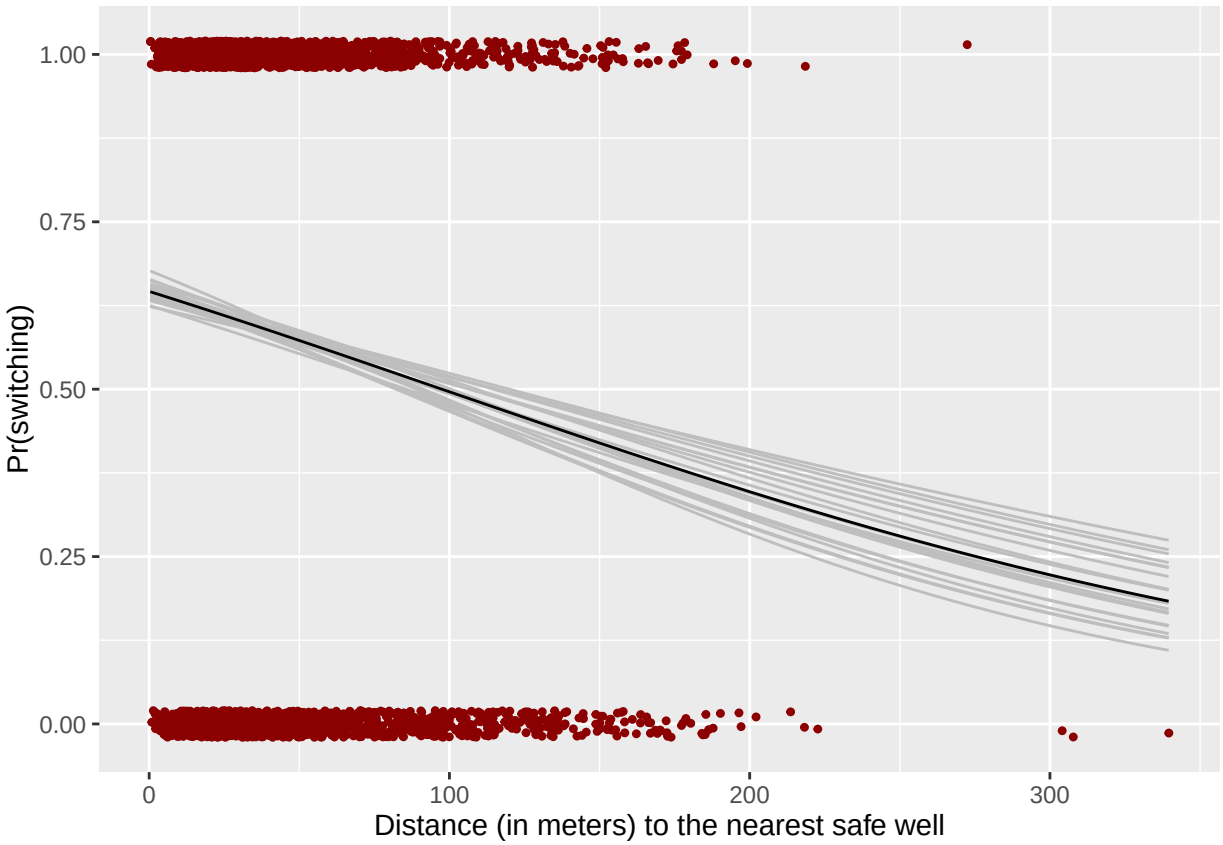
```
fit1.post <- extract(wells.sf1)
beta.mean <- colMeans(fit1.post$beta)
```

figure 7.6

```
# figure 7.6a
frame1 <- data.frame(x1 = fit1.post$beta[,1], y1 = fit1.post$beta[,2])
p1 <- ggplot(frame1, aes(x1, y1)) +
     geom_point(shape = 20, color = 'darkred') +
     theme_gray() +
     scale_y_continuous(expression(beta[1])) +
     scale_x_continuous(expression(beta[0]))
print(p1)
```




```
 # Figure 7.6 (b)
# dev.new()
# changed distances to 1 m instead of 100 m
# removed 4000- from fit1.post$beta[4000-",i,",2]
frame2 = data.frame(x1=dist,y1=switc)
p2 <- "ggplot(frame2,aes(x=x1,y=y1)) +
       geom_jitter(position = position_jitter(width = 0.2, height = 0.05),
         shape = 20, color = 'darkred') +
       theme_gray() +
       scale_y_continuous('Pr(switching)') +
       scale_x_continuous('Distance (in meters) to the nearest safe well')"
for (i in 1:20) {
  p2 <- paste(p2,"+ stat_function(aes(y=0),fun=function(x) 1.0 / (1 + exp(-fit1.post$beta[",i,",1]-fit1
}
p2 <- paste(p2, "+ stat_function(fun=function(x) 1.0 / (1 + exp(-beta.mean[1] - beta.mean[2] * x)))")
eval(parse(text = p2))
```



predictive simulation using the binomial distribution

```
## Predictive simulation using the binomial distribution
n.sims <- 4000
X.tilde <- cbind (1, dist)
n.tilde <- nrow (X.tilde)
y.tilde <- array (NA, c(n.sims, n.tilde))
# throws error : subscript out of bounds
# n.sims = 4000, but fit1.post$beta has 2000 entries
# for (s in 1:n.sims){
for (s in 1:2000){
  p.tilde <- invlogit (X.tilde %*% fit1.post$beta[s,])
  y.tilde <- rbinom (n.tilde, 1, p.tilde)
}
```

predictive simulation using latent logistic distribution

```
## Predictive simulation using latent logistic distribution
logit <- function (a) {log(a/(1-a))}

y.tilde <- array (NA, c(n.sims, n.tilde))
# throws error : subscript out of bounds
```

```
# n.sims = 4000, but fit1.post$beta has 2000 entries
# for (s in 1:n.sims){
for (s in 1:2000){
  epsilon.tilde <- logit (runif (n.tilde, 0, 1))
  z.tilde <- X.tilde %*% fit1.post$beta[s,] + epsilon.tilde
  y.tilde[s,] <- ifelse (z.tilde>0, 1, 0)
}
```

alternative using matrix algebra

```
# Alternative using matrix algebra
## Error in fit1.post$beta %*% t(X.tilde) + epsilon.tilde :
##   non-conformable arrays
# epsilon.tilde <- array (logit (runif (n.sims*n.tilde, 0, 1)),
#                         c(n.sims, n.tilde))
# z.tilde <- fit1.post$beta %*% t(X.tilde) + epsilon.tilde
# y.tilde <- ifelse (z.tilde>0, 1, 0)
```

compound models

earnings1

data

```
### Compound models

## Models (earnings1.stan)
## glm (earn_pos ~ height + male, family=binomial(link="logit"))
source("earnings1.data.R", echo = TRUE)
```

```
##
## > "earn_pos" <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
## +     1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
## +     0, 1, 0, 1, 0, 1, 1, 1, 1 .... [TRUNCATED]
##
## > "height" <- c(74, 66, 64, 63, 63, 64, 62, 73, 72,
## +     72, 70, 63, 68, 68, 65, 60, 66, 68, 68, 70, 67, 64, 73, 62,
## +     63, 67, 66, 72, 63, 68,  .... [TRUNCATED]
##
## > "male" <- c(1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1,
## +     1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0,
## +     1, 0, 1, 0, 0, 0, 1, 1, 0, .... [TRUNCATED]
##
## > "N" <- 1379
```

model

earnings1.stan

```
data {
  int<lower=0> N;
  int<lower=0,upper=1> earn_pos[N];
  vector[N] height;
  vector[N] male;
}
parameters {
  vector[3] beta;
  real<lower=0> sigma;
}
model {
  earn_pos ~ bernoulli_logit(beta[1] + beta[2] * height + beta[3] * male);
}
```

fit

```
dataList.2 <- c("N","earn_pos","height","male")
earnings1.sf1 <- stan(file='earnings1.stan', data=dataList.2,
                      iter=1000, chains=4)
```

```
## Warning: There were 1791 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
print(earnings1.sf1)
```

```
## Inference for Stan model: earnings1.
## 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%
## beta[1]  -2.740000e+00    0.97 1.38  -4.980000e+00  -3.450000e+00
## beta[2]   6.000000e-02    0.02 0.02   4.000000e-02   5.000000e-02
## beta[3]   1.950000e+00    0.14 0.20   1.510000e+00   1.870000e+00
## sigma    8.870381e+307     Inf  Inf  2.623713e+306  4.007646e+307
## lp__      2.126800e+02    0.17 1.36   2.091300e+02   2.120400e+02
##                    50%            75%          97.5% n_eff  Rhat
## beta[1]  -2.480000e+00  -1.770000e+00  -1.070000e+00     2 13.36
## beta[2]   6.000000e-02   8.000000e-02   1.000000e-01     2 10.80
## beta[3]   2.010000e+00   2.110000e+00   2.160000e+00     2  6.24
## sigma    8.996511e+307  1.362923e+308  1.766087e+308  2000   NaN
## lp__      2.130000e+02   2.136400e+02   2.143900e+02    61  1.05
##
## Samples were drawn using NUTS(diag_e) at Thu Jul  7 13:22:42 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 1
```

```
fit1a.post <- extract(earnings1.sf1)
```

earnings2

data

```
## (earnings2.stan)
##model lm (log.earn ~ height + male, subset=earnings>0)
source("earnings2.data.R", echo = TRUE)
```

```
##
## > "earnings" <- c(50000, 60000, 30000, 50000, 51000,
## +     9000, 29000, 32000, 2000, 27000, 6530, 30000, 12000, 12000,
## +     22000, 17000, 40000, 44 .... [TRUNCATED]
##
## > "height1" <- c(6, 5, 5, 5, 5, 5, 5, 6, 6, 6, 5, 5,
## +     5, 5, 5, 5, 5, 5, 5, 6, 5, 5, 5, 5, 6, 5, 5, 6, 6, 5, 5,
## +     6, 5, 5, 5, 5, 5, 5, 5, 5, .... [TRUNCATED]
##
## > "height2" <- c(2, 6, 4, 3, 3, 4, 2, 1, 0, 0, 10, 8,
## +     8, 5, 6, 8, 8, 10, 4, 1, 2, 3, 7, 6, 0, 3, 4, 0, 5, 4, 4,
## +     0, 8, 4, 1, 7, 5, 4, 7,  .... [TRUNCATED]
##
## > "sex" <- c(1, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1,
## +     2, 2, 1, 2, 1, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 1, 2, 2, 1,
## +     1, 2, 2, 2, 2, 2, 2, 2, 2,  .... [TRUNCATED]
##
## > "race" <- c(1, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## +     1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1,
## +     1, 1, 1, 2, 1, 1, 1, 1, 1, .... [TRUNCATED]
##
## > "hispanic" <- c(2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2,
## +     2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
## +     2, 2, 2, 2, 2, 2, 2, 2, 2 .... [TRUNCATED]
##
## > "education" <- c(16, 16, 16, 16, 17, 15, 12, 17, 15,
## +     12, 16, 11, 12, 12, 16, 12, 14, 13, 12, 13, 13, 14, 14, 12,
## +     12, 12, 16, 16, 16, 1 .... [TRUNCATED]
##
## > "yearbn" <- c(45, 32, 61, 99, 51, 64, 41, 44, 69,
## +     64, 25, 56, 63, 39, 55, 32, 61, 46, 35, 55, 39, 69, 68, 49,
## +     45, 55, 30, 52, 57, 65,  .... [TRUNCATED]
##
## > "height" <- c(74, 66, 64, 63, 63, 64, 62, 73, 72,
## +     72, 70, 68, 68, 65, 66, 68, 68, 70, 64, 73, 62, 63, 67, 66,
## +     72, 63, 64, 72, 77, 64,  .... [TRUNCATED]
##
## > "N" <- 1192
```

model

earnings2.stan

```
data {
  int<lower=0> N;
```

```
  vector[N] earnings;
  vector[N] height;
  vector[N] sex;
}
transformed data {
  vector[N] log_earnings;
  vector[N] male;

  log_earnings = log(earnings);
  male = 2 - sex;
}
parameters {
  vector[3] beta;
  real<lower=0> sigma;
}
model {
  log_earnings ~ normal(beta[1] + beta[2] * height + beta[3] * male, sigma);
}
```

fit

```
dataList.3 <- c("N","earnings","height","sex")
earnings2.sf1 <- stan(file='earnings2.stan', data=dataList.3,
                      iter=1000, chains=4)
print(earnings2.sf1)
```

```
## Inference for Stan model: earnings2.
## 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% n_eff
## beta[1]    8.10    0.03 0.61    6.89    7.68    8.09    8.50    9.33   391
## beta[2]    0.02    0.00 0.01    0.00    0.02    0.02    0.03    0.04   387
## beta[3]    0.42    0.00 0.08    0.28    0.37    0.42    0.47    0.57   577
## sigma      0.88    0.00 0.02    0.85    0.87    0.88    0.89    0.91   706
## lp__    -445.55    0.06 1.47 -449.21 -446.28 -445.18 -444.48 -443.73   567
##         Rhat
## beta[1] 1.01
## beta[2] 1.01
## beta[3] 1.00
## sigma   1.01
## lp__    1.01
##
## Samples were drawn using NUTS(diag_e) at Thu Jul  7 13:22:53 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
##  0.9 1 1 1.3
```

```
fit1b.post <- extract(earnings2.sf1)

x.new <- c (1, 68, 1)          # constant term=1, height=68, male=1
```

simulation ignoring uncertainty

```
 # Simulation ignoring uncertainty
n.sims <- 4000
prob.earn.pos <- invlogit (fit1a.post$beta %*% x.new)
earn.pos.sim <- rbinom (n.sims, 1, prob.earn.pos)
earn.sim <- ifelse (earn.pos.sim==0, 0,
  exp (rnorm (n.sims, fit1a.post$beta %*% x.new,mean(fit1b.post$sigma))))
```

simulated values of coefficient estimates

```
 # Simulated values of coefficient estimates
prob.earn.pos <- invlogit (fit1a.post$beta %*% x.new)
earn.pos.sim <- rbinom (n.sims, 1, prob.earn.pos)
earn.sim <- ifelse (earn.pos.sim==0, 0,
  exp (rnorm (n.sims, fit1b.post$beta %*% x.new, fit1b.post$sigma)))
```

computations into a function

```
# Computations into a function
Mean.earn <- function (height, male, fit1a.post, fit1b.post){
  x.new <- c (1, height, male)
  prob.earn.pos <- invlogit (fit1a.post$beta%*% x.new)
  earn.pos.sim <- rbinom (n.sims, 1, prob.earn.pos)
  earn.sim <- ifelse (earn.pos.sim==0, 0,
    exp (rnorm (n.sims, fit1b.post$beta %*% x.new, fit1b.post$sigma)))
  return (mean (earn.sim))
}

heights <- seq (60, 75, 1)
mean.earn.female <- sapply (heights, Mean.earn, male=0, fit1a.post, fit1b.post)
mean.earn.male <- sapply (heights, Mean.earn, male=1, fit1a.post, fit1b.post)
```

figure 7.8

```
# dev.new()
frame <- data.frame(x1 = heights, x2 = mean.earn.female, x3 = mean.earn.male)
p3 <- ggplot(frame,aes(x1, x2, x3)) +
      geom_line(aes(y=x2, x=x1), color = "steelblue") +
      geom_point(aes(y=x2, x=x1), shape = 20, color = "darkred") +
      geom_line(aes(y=x3, x=x1), color = "steelblue") +
```

```
      geom_point(aes(y=x3, x=x1), shape = 20, color = "darkred") +
      annotate("text", x = c(72, 72), y = c(16500, 30500), label = c("women", "men")) +
      labs(x = "height", y = "mean predicted earnings", title = "n.sims = 4000") +
      theme_gray()
print(p3)
```