8.1 fake data simulation

Chris Parrish

July 3, 2016

Contents

8.1 fake data simulation

reference:
- ARM chapter 08, github

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

fake-data simulation

1. select 'true' values of parameters a, b, sigma
2. generate data set x, y : y ~ a + b * x + epsilon
3. fit y ~ x given data
4. check to see that true values are within 1 or 2 SEs of the fitted parameters

```
## Fake-data simulation
a <- 1.4
b <- 2.3
sigma <- 0.9
x <- 1:5
n <- length(x)
```

simulate data, fit the model, and check the coverage of the conf intervals

```
# Simulate data, fit the model, and check the coverage of the conf intervals
y <- a + b*x + rnorm (n, 0, sigma)
```

model

y__x.stan

```
data {
  int<lower=0> N;
  vector[N] x;
  vector[N] y;
}
parameters {
  vector[2] beta;
  real<lower=0> sigma;
}
model {
  y ~ normal(beta[1] + beta[2] * x, sigma);
}
```

fit

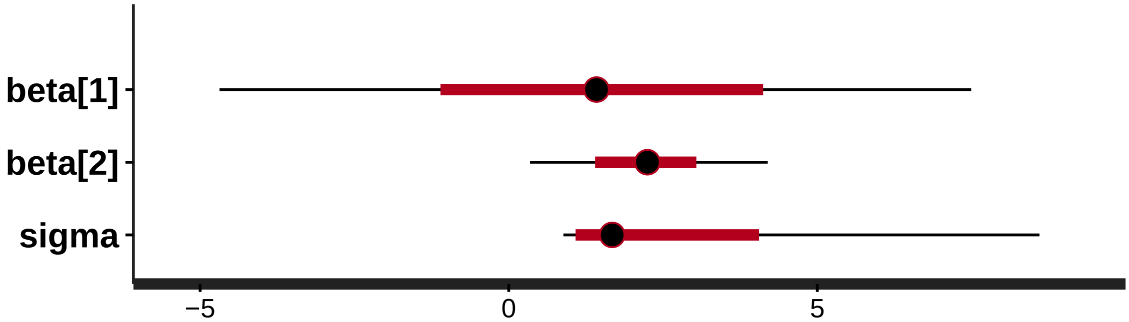
```r
# (y_x.stan)
# lm(y ~ x)
dataList.1 <- list(N=length(y), y=y, x=x)
y_x.sf1 <- stan(file='y_x.stan', data=dataList.1, iter=1000, chains=4)
```

```
## Warning: There were 282 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

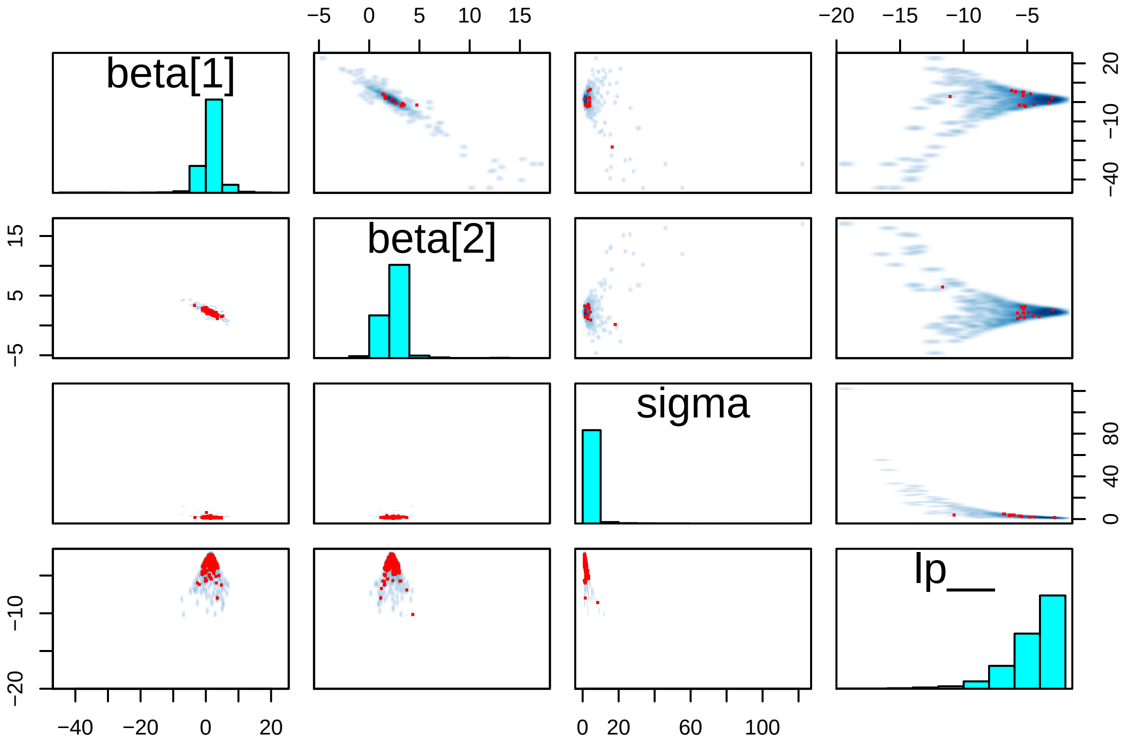
```r
y_x.sm <- y_x.sf1@stanmodel         # extract stanmodel
```

```r
plot(y_x.sf1)
```

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

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




```r
pairs(y_x.sf1)
```




```r
print(y_x.sf1)
```

```
## Inference for Stan model: y_x.
## 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 Rhat
## beta[1]  1.22    0.42 4.08 -4.69  0.11  1.42  2.52  7.49    93 1.03
## beta[2]  2.29    0.13 1.24  0.35  1.89  2.25  2.60  4.20    91 1.04
## sigma    2.51    0.31 4.01  0.88  1.31  1.68  2.54  8.59   170 1.02
## lp__    -4.58    0.18 2.09 -9.89 -5.54 -3.97 -3.08 -2.29   131 1.01
##
## Samples were drawn using NUTS(diag_e) at Fri Jul  8 01:33:47 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
##  0.6 0.6 0.7 0.3
```

```r
post <- extract(y_x.sf1)
b.hat <- colMeans(post$beta)[2]           # "b" is the 2nd coef in the model
b.se <- sd(post$beta[,2]) / sqrt(4000)      # "b" is the 2nd coef in the model
rbind(b.hat = b.hat, b.se = b.se)
```

```
##             [,1]
## b.hat 2.29342854
## b.se  0.01954221
```

```r
cover.68 <- abs (b - b.hat) < b.se      # this will be TRUE or FALSE
cover.95 <- abs (b - b.hat) < 2*b.se    # this will be TRUE or FALSE
```

```r
cat (paste ("68% coverage: ", cover.68, "\n"))
```

```
## 68% coverage:  TRUE
```

```r
cat (paste ("95% coverage: ", cover.95, "\n"))
```

```
## 95% coverage:  TRUE
```

put it in a loop

```r
# Put it in a loop
# n.fake <- 1000
n.fake <- 10
cover.68 <- rep (NA, n.fake)
cover.95 <- rep (NA, n.fake)
for (s in 1:n.fake){
  y <- a + b*x + rnorm (n, 0, sigma)
  dataList.1 <- list(N=length(y), y=y, x=x)
  y_x.sf1 <- sampling(y_x.sm, dataList.1)      # sampling is in rstan
#  print(y_x.sf1)
  post <- extract(y_x.sf1)
  b.hat <- colMeans(post$beta)[2]            # "b" is the 2nd coef in the model
  b.se <- sd(post$beta[,2]) / sqrt(4000)     # "b" is the 2nd coef in the model
#  print(rbind(b.hat = b.hat, b.se = b.se))
  cover.68[s] <- abs (b - b.hat) < b.se
  cover.95[s] <- abs (b - b.hat) < 2*b.se
}
```

```
## Warning: There were 210 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

## Warning: There were 350 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

## Warning: There were 234 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

## Warning: There were 337 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

## Warning: There were 296 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

## Warning: There were 238 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

```
## Warning: There were 415 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

## Warning: There were 58 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

## Warning: There were 664 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

## Warning: There were 339 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

```
cat (paste ("68% coverage: ", mean(cover.68), "\n"))
```

```
## 68% coverage:  0
```

```
cat (paste ("95% coverage: ", mean(cover.95), "\n"))
```

```
## 95% coverage:  0.1
```

do it again, this time using t intervals

```
# Do it again, this time using t intervals
# n.fake <- 1000
n.fake <- 10
cover.68 <- rep (NA, n.fake)
cover.95 <- rep (NA, n.fake)
t.68 <-  qt (.84, n-2)
t.95 <-  qt (.975, n-2)
for (s in 1:n.fake){
  y <- a + b*x + rnorm (n, 0, sigma)
  dataList.1 <- list(N=length(y), y=y, x=x)
  y_x.sf1 <- sampling(y_x.sm, dataList.1)
#  print(y_x.sf1)
  post <- extract(y_x.sf1)
  b.hat <- colMeans(post$beta)[2]            # "b" is the 2nd coef in the model
  b.se <- sd(post$beta[,2]) / sqrt(4000)     # "b" is the 2nd coef in the model
#  print(rbind(b.hat = b.hat, b.se = b.se))
  cover.68[s] <- abs (b - b.hat) < t.68*b.se
  cover.95[s] <- abs (b - b.hat) < t.95*b.se
}
```

```
## Warning: There were 9 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

## Warning: There were 233 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

```
## Warning: There were 199 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

## Warning: There were 316 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

## Warning: There were 326 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

## Warning: There were 103 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

## Warning: There were 210 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

## Warning: There were 170 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

## Warning: There were 68 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

## Warning: There were 279 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help.

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

```
cat (paste ("68% coverage: ", mean(cover.68), "\n"))
```

```
## 68% coverage:  0.2
```

```
cat (paste ("95% coverage: ", mean(cover.95), "\n"))
```

```
## 95% coverage:  0.3
```

do it again, this time using t intervals and lm to fit the model

```
# Do it again, this time using t intervals
# n.fake <- 1000
n.fake <- 1000
cover.68 <- rep (NA, n.fake)
cover.95 <- rep (NA, n.fake)
t.68 <-  qt (.84, n-2)
t.95 <-  qt (.975, n-2)
for (s in 1:n.fake){
  y <- a + b*x + rnorm (n, 0, sigma)
  lm1 <- lm(y ~ x)
  b.hat <- coef(lm1)[2]
```

```r
  b.se <- se.coef(lm1)[2]
  cover.68[s] <- abs (b - b.hat) < t.68*b.se
  cover.95[s] <- abs (b - b.hat) < t.95*b.se
}
cat (paste ("68% coverage: ", mean(cover.68), "\n"))
```

```
## 68% coverage:  0.675
```

```r
cat (paste ("95% coverage: ", mean(cover.95), "\n"))
```

```
## 95% coverage:  0.954
```