Outline (April 8, 2014 – DJC)

1. Compared posteriors in a 1D probit regression example: Phil's SIR and Lynne's Stan code. (Result: VERY similar) I did those by plotting the quantiles, although I also like Phil's suggestion ("plot two large sorted samples against each other and see if they match the y=x line)"), which I haven't done yet.
2. Extended Lynne's Stan code to work on multiple probit regression.
3. Fit an example compare posteriors in a 2D probit example (intercept + one feature) for the Stan code vs. SIR. I only compared for one parameter, but the quantile plot for Stan and SIR were still quite similar (maybe not quite as similar as in the 1D case, though).

```
library(rstan)
```

```
## Loading required package: Rcpp
## Loading required package: inline
## Loading required package: methods
##
## Attaching package: 'inline'
##
## The following object is masked from 'package:Rcpp':
##
##     registerPlugin
##
## rstan (Version 2.2.0, packaged: 2014-02-14 04:29:17 UTC, GitRev: 52d7b230aaa0)
```

```
library(ggplot2)
library(coda)
```

```
## Loading required package: lattice
##
## Attaching package: 'coda'
##
## The following object is masked from 'package:rstan':
##
##     traceplot
```

```
library(plyr)
source("Sir/drawbetap.R")
set.seed(908952)
```

Comparing Stan and SIR in 1D Probit Example

First we simulate some data (one predictor probit, no intercept):

```
nStanIterations <- 3000

n = 1000
beta <- 0.5

# sample from the probit model via latent data
x = rnorm(n, 0, 1)
error <- rnorm(n, 0, 1)
z <- x * beta + error
y <- as.integer(z > 0)
```

Now fit via Stan and SIR:

```
s71probit_dat <- list(Y = y, x = x, n = 1000)
stanFitProbit1D <- stan("Stan/Probit1D.Stan", data = s71probit_dat, iter = nStanIterations,
    chains = 4)
```

```
##
## TRANSLATING MODEL 'Probit1D' FROM Stan CODE TO C++ CODE NOW.
## COMPILING THE C++ CODE FOR MODEL 'Probit1D' NOW.
## SAMPLING FOR MODEL 'Probit1D' NOW (CHAIN 1).
## Iteration:    1 / 3000 [  0%]  (Warmup)Iteration:  300 / 3000 [ 10%]  (Warmup)Iteration:
## Elapsed Time: 5.72803 seconds (Warm-up)
##               5.57056 seconds (Sampling)
##               11.2986 seconds (Total)
##
## SAMPLING FOR MODEL 'Probit1D' NOW (CHAIN 2).
## Iteration:    1 / 3000 [  0%]  (Warmup)Iteration:  300 / 3000 [ 10%]  (Warmup)Iteration:
## Elapsed Time: 5.80312 seconds (Warm-up)
##               5.56637 seconds (Sampling)
##               11.3695 seconds (Total)
##
## SAMPLING FOR MODEL 'Probit1D' NOW (CHAIN 3).
## Iteration:    1 / 3000 [  0%]  (Warmup)Iteration:  300 / 3000 [ 10%]  (Warmup)Iteration:
## Elapsed Time: 5.73923 seconds (Warm-up)
##               5.58587 seconds (Sampling)
##               11.3251 seconds (Total)
##
## SAMPLING FOR MODEL 'Probit1D' NOW (CHAIN 4).
## Iteration:    1 / 3000 [  0%]  (Warmup)Iteration:  300 / 3000 [ 10%]  (Warmup)Iteration:
## Elapsed Time: 5.57332 seconds (Warm-up)
##                6.22901 seconds (Sampling)
##                11.8023 seconds (Total)
```

```
stanFitProbit1D
```

```
## Inference for Stan model: Probit1D.
## 4 chains, each with iter=3000; warmup=1500; thin=1;
## post-warmup draws per chain=1500, total post-warmup draws=6000.
##
##         mean se_mean  sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## beta1    0.5       0 0.0    0.4    0.5    0.5    0.5    0.6  2152    1
## lp__  -622.9       0 0.7 -624.9 -623.1 -622.7 -622.5 -622.4  2521    1
##
## Samples were drawn using NUTS(diag_e) at Tue Apr  8 23:23:14 2014.
## 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).
```

```
betaSamplesStan <- extract(stanFitProbit1D)$beta1
betaSamplesSir <- drawbeta(5000, matrix(x), y)
```

```
## 10000 , 20000 , 30000 , 40000 , 50000 ,
```

```
samplesDF <- rbind(data.frame(beta = betaSamplesStan, sampler = "STAN"), data.frame(beta = b
    sampler = "SIR"))
```

Here’s a histogram of the samples:

```
ggplot(samplesDF) + geom_histogram(aes(x = beta, y = ..density..)) + facet_grid(sampler ~
    .) + ggtitle("Histogram of Samples Under Each Method")
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```

Here’s a plot of the quantiles. The distributions are very similar! (Note that the MCMC samples do not need to be thinned for this purpose. Autocorrelated samples are fine, as long as the sampler converged to the right distribution and we have enough of them. From the Stan diagnostics above, both of these are true. Thinning would just throw away information.)

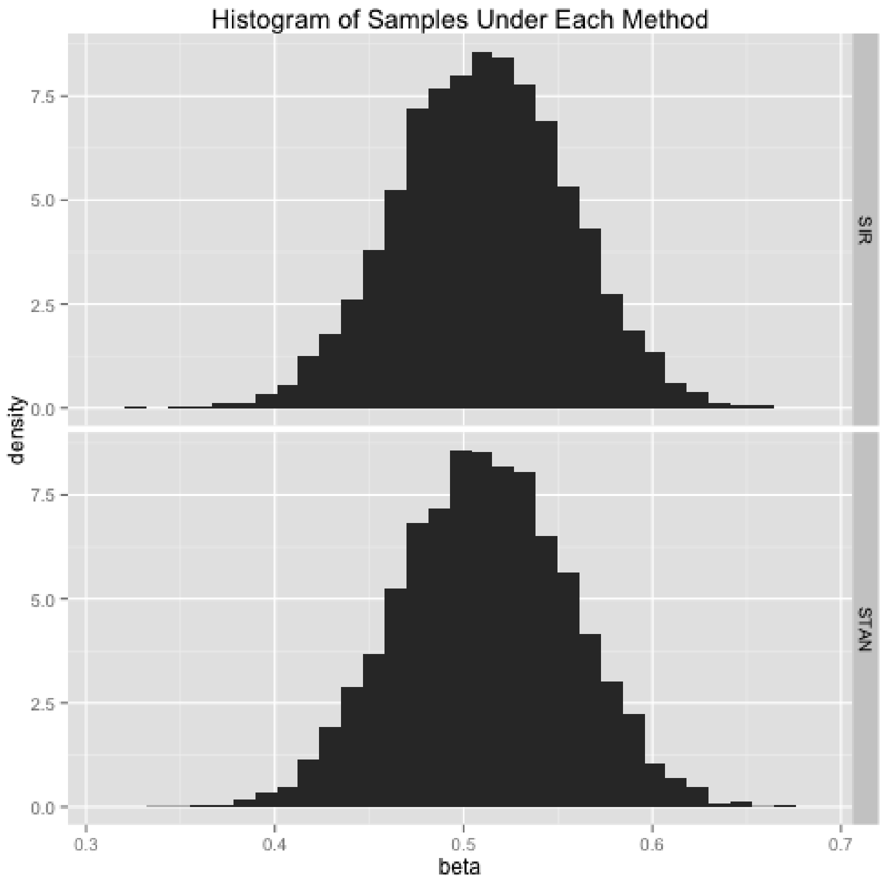

Figure 1: plot of chunk unnamed-chunk-4

```
probs <- seq(0, 1, by = 0.01)
quantilesDF <- ddply(samplesDF, "sampler", function(df) {
    return(data.frame(prob = probs, quantile = quantile(df$beta, probs = probs)))
})

ggplot(quantilesDF) + geom_point(aes(x = prob, y = quantile, color = sampler),
    alpha = 0.5, size = 3) + ggtitle("Quantiles of the Samples Under Each Method")
```

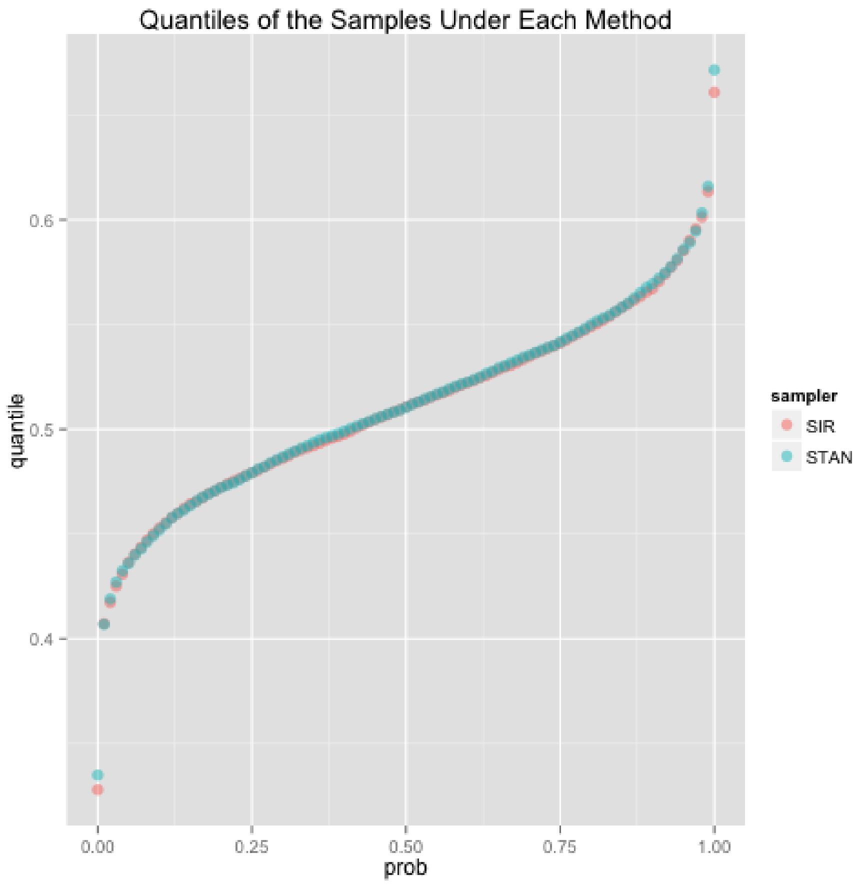

Figure 2: plot of chunk unnamed-chunk-5

Probit with Intercept

Create simulated data:

```r
n = 1000
x1 = rep(1, n)  # intercept
x2 = runif(n, min = -2, max = 2)
X <- cbind(x1, x2)
beta <- c(0.5, 1)
error <- rnorm(n, 0, 1)
z <- (X %*% beta) + error
y <- as.integer(z > 0)
```

Now fit via Stan and SIR:

```r
stanFitProbitMultiple <- stan("Stan/ProbitMultipleRegression.Stan", data = list(Y = y,
    X = X, n = n, m = 2), iter = nStanIterations, chains = 4)
```

```
##
## TRANSLATING MODEL 'ProbitMultipleRegression' FROM Stan CODE TO C++ CODE NOW.
## COMPILING THE C++ CODE FOR MODEL 'ProbitMultipleRegression' NOW.
## SAMPLING FOR MODEL 'ProbitMultipleRegression' NOW (CHAIN 1).
## Iteration:    1 / 3000 [  0%]  (Warmup)Iteration:  300 / 3000 [ 10%]  (Warmup)Iteration:
## Elapsed Time: 16.1835 seconds (Warm-up)
##               18.7993 seconds (Sampling)
##               34.9828 seconds (Total)
##
## SAMPLING FOR MODEL 'ProbitMultipleRegression' NOW (CHAIN 2).
## Iteration:    1 / 3000 [  0%]  (Warmup)Iteration:  300 / 3000 [ 10%]  (Warmup)Iteration:
## Elapsed Time: 15.8844 seconds (Warm-up)
##               17.7737 seconds (Sampling)
##               33.6581 seconds (Total)
##
## SAMPLING FOR MODEL 'ProbitMultipleRegression' NOW (CHAIN 3).
## Iteration:    1 / 3000 [  0%]  (Warmup)Iteration:  300 / 3000 [ 10%]  (Warmup)Iteration:
## Elapsed Time: 16.0195 seconds (Warm-up)
##               21.0521 seconds (Sampling)
##               37.0716 seconds (Total)
##
## SAMPLING FOR MODEL 'ProbitMultipleRegression' NOW (CHAIN 4).
## Iteration:    1 / 3000 [  0%]  (Warmup)Iteration:  300 / 3000 [ 10%]  (Warmup)Iteration:
## Elapsed Time: 16.4983 seconds (Warm-up)
##               17.7643 seconds (Sampling)
##               34.2625 seconds (Total)
```

```r
stanFitProbitMultiple
```

```
## Inference for Stan model: ProbitMultipleRegression.
## 4 chains, each with iter=3000; warmup=1500; thin=1;
## post-warmup draws per chain=1500, total post-warmup draws=6000.
##
##           mean se_mean  sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## beta[1]    0.5       0 0.1    0.4    0.5    0.5    0.6    0.6  2854    1
## beta[2]    1.1       0 0.1    0.9    1.0    1.1    1.1    1.2  2540    1
## lp__    -391.7       0 1.1 -394.6 -392.0 -391.3 -390.9 -390.7  1252    1
##
## Samples were drawn using NUTS(diag_e) at Tue Apr  8 23:25:52 2014.
## 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).
```

```
betaSamplesStan <- extract(stanFitProbitMultiple)$beta
beta1SamplesStan <- betaSamplesStan[, 2]

betaSamplesSir <- drawbeta(5000, X, y)
```

```
## 10000 , 20000 , 30000 , 40000 , 50000 ,
```

```
beta1SamplesSir <- betaSamplesSir[2, ]

samplesDF <- rbind(data.frame(beta1 = beta1SamplesStan, sampler = "STAN"), data.frame(beta1
    sampler = "SIR"))
```

I'm only comparing the posteriors for the intercept parameter for now:

```
ggplot(samplesDF) + geom_histogram(aes(x = beta1, y = ..density..)) + facet_grid(sampler ~
    .) + ggtitle("Histogram of Samples Under Each Method")
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.

## Warning: position_stack requires constant width: output may be incorrect
## Warning: position_stack requires constant width: output may be incorrect
```

Here's a plot of the quantiles. Still pretty similar.

```
probs <- seq(0, 1, by = 0.01)
quantilesDF <- ddply(samplesDF, "sampler", function(df) {
    return(data.frame(prob = probs, quantile = quantile(df$beta1, probs = probs)))
})

ggplot(quantilesDF) + geom_point(aes(x = prob, y = quantile, color = sampler),
    alpha = 0.5, size = 3) + ggtitle("Quantiles of the Samples Under Each Method")
```

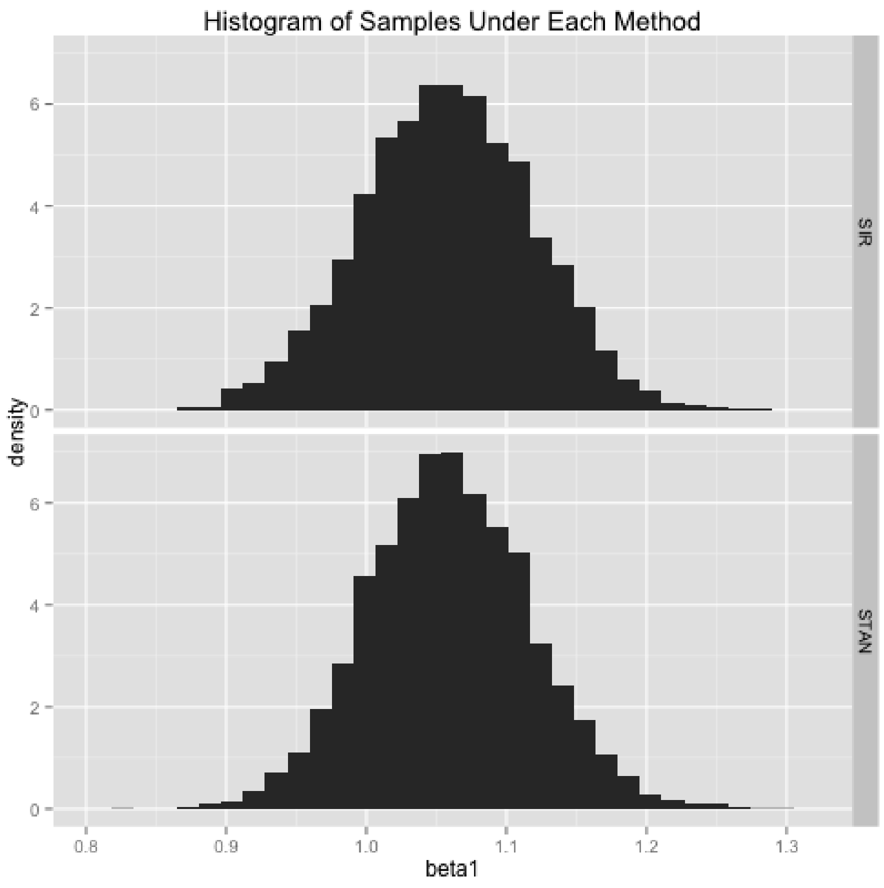

Figure 3: plot of chunk unnamed-chunk-8

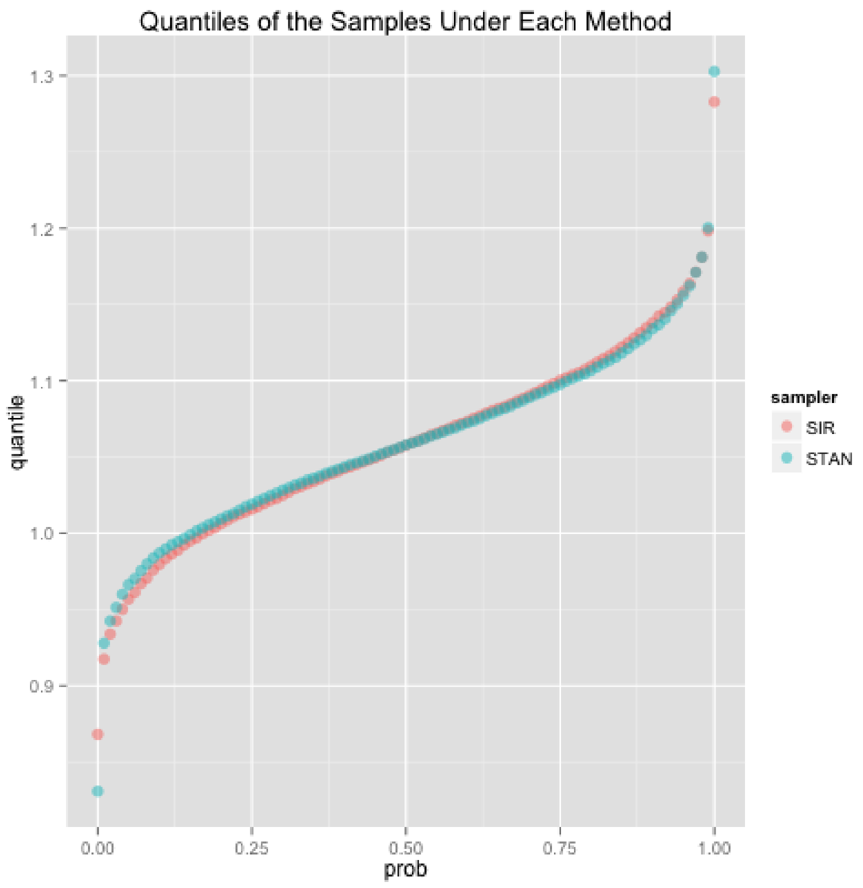

Figure 4: plot of chunk unnamed-chunk-9