lp__ in Stan output

Xulong Wang

October 20, 2015

```r
library(rstan)
```

```
## Warning: package 'rstan' was built under R version 3.1.3
```

```
## Loading required package: Rcpp
```

```
## Warning: package 'Rcpp' was built under R version 3.1.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.1.3
```

```
## rstan (Version 2.8.0, packaged: 2015-09-19 14:48:38 UTC, GitRev: 05c3d0058b6a)
## For execution on a local, multicore CPU with excess RAM we recommend calling
## rstan_options(auto_write = TRUE)
## options(mc.cores = parallel::detectCores())
```

```r
stan_code <- "

data {
  int<lower=0> N;
  real y[N];
}

parameters {
  real mu;
  real<lower=machine_precision()> sigma;
}

model {
  y ~ normal(mu, sigma);
}

generated quantities {
  real lpd;
  lpd <- normal_log(y, mu, sigma);
}

"

dat <- list(N = 20, y = rnorm(20, 1, 1)) # pseudo-data

model <- stan_model(model_code = stan_code) # compilation
myfit <- sampling(model, data = dat) # fit by sampling
```

```
##
## SAMPLING FOR MODEL 'a617ae501b9cd3408045b59a635fed62' NOW (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)
## #  Elapsed Time: 0.010969 seconds (Warm-up)
## #                0.010779 seconds (Sampling)
## #                0.021748 seconds (Total)
##
##
## SAMPLING FOR MODEL 'a617ae501b9cd3408045b59a635fed62' NOW (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)
## #  Elapsed Time: 0.011276 seconds (Warm-up)
## #                0.011163 seconds (Sampling)
## #                0.022439 seconds (Total)
##
##
## SAMPLING FOR MODEL 'a617ae501b9cd3408045b59a635fed62' NOW (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)
## #  Elapsed Time: 0.011691 seconds (Warm-up)
## #                0.011245 seconds (Sampling)
## #                0.022936 seconds (Total)
##
##
## SAMPLING FOR MODEL 'a617ae501b9cd3408045b59a635fed62' NOW (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)
## #  Elapsed Time: 0.012901 seconds (Warm-up)
## #                0.012573 seconds (Sampling)
## #                0.025474 seconds (Total)
```

```
print(myfit)
```

```
## Inference for Stan model: a617ae501b9cd3408045b59a635fed62.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##         mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## mu      0.80    0.01 0.19   0.43   0.68   0.80   0.93   1.17  1352 1.00
## sigma   0.83    0.00 0.15   0.60   0.73   0.81   0.92   1.18  1252 1.00
## lpd   -23.92    0.04 1.13 -26.93 -24.37 -23.56 -23.12 -22.81   779 1.01
## lp__   -5.74    0.04 1.06  -8.51  -6.17  -5.41  -4.99  -4.69   795 1.01
##
## Samples were drawn using NUTS(diag_e) at Tue Oct 20 13:00:07 2015.
## 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).
```

```
# print() shows summary statistics of model parameters, with an extra "lp__" term at the bottom.

y = extract(myfit) # samples

my_lpd = sapply(1:4000, function(i) sum(dnorm(dat$y, mean = y$mu[i], sd = y$sigma[i], log = TRUE))) # t

cons = log(1/sqrt(2 * pi)) # the constant in normal distribution
my_lp__ = my_lpd - 20 * cons # Stan's lp__

head(y$lpd, n = 30) # retrieve true lp in Stan
```

```
##  [1] -23.23121 -23.12710 -23.74555 -23.15083 -24.20991 -23.77152 -24.41861
##  [8] -24.98320 -23.23947 -24.94968 -23.24113 -25.17634 -23.70241 -23.79789
## [15] -24.00484 -22.79008 -24.06494 -22.78593 -24.61845 -25.27698 -23.04777
## [22] -25.73212 -23.03347 -23.98554 -23.82947 -25.48557 -25.58914 -23.29489
## [29] -22.97151 -23.72995
```

```
head(my_lpd, n = 30)
```

```
##  [1] -23.23121 -23.12710 -23.74555 -23.15083 -24.20991 -23.77152 -24.41861
##  [8] -24.98320 -23.23947 -24.94968 -23.24113 -25.17634 -23.70241 -23.79789
## [15] -24.00484 -22.79008 -24.06494 -22.78593 -24.61845 -25.27698 -23.04777
## [22] -25.73212 -23.03347 -23.98554 -23.82947 -25.48557 -25.58914 -23.29489
## [29] -22.97151 -23.72995
```

```
head(y$lp__, n = 30) # Stan's "fake" lp
```

```
##  [1] -5.076629 -4.902121 -5.504755 -5.150019 -6.336471 -5.475901 -6.289782
##  [8] -6.968047 -5.283797 -6.944664 -5.014412 -7.009417 -5.491333 -5.815071
## [15] -5.924474 -4.694721 -5.695792 -4.692545 -6.274036 -7.073440 -5.054612
## [22] -7.795791 -5.036533 -5.760120 -5.522156 -7.244012 -7.098111 -5.179818
## [29] -4.869267 -5.583251
```

```
head(my_lp__, n = 30)
```

```
##  [1] -4.852444 -4.748330 -5.366775 -4.772064 -5.831142 -5.392752 -6.039836
##  [8] -6.604433 -4.860704 -6.570913 -4.862361 -6.797569 -5.323641 -5.419123
## [15] -5.626066 -4.411305 -5.686165 -4.407158 -6.239681 -6.898212 -4.669002
## [22] -7.353346 -4.654698 -5.606768 -5.450698 -7.106804 -7.210366 -4.916116
## [29] -4.592743 -5.351182
```