non_invertible_varying_alpha.r

max

Thu Feb 8 18:49:23 2018

```
### This script implements example from page 1250 in Poole & Raftery (2000), JASA
### Original code by Gabriel Mendes (Berkeley): http://discourse.mc-stan.org/t/bayesian-melding/3011
### Implements the (unormalised) exact target of the example, mainly to demonstrate correctness
##### Copyleft (or the one to blame): Luiz Max Carvalho (2018)
varying_alpha <- '
functions{
real fZ_exact_lpdf(real z, real ax, real bx, real ay, real by){
// notice the lack of in-built check: ay/bx < x < by/ax
real k;
real L;
real U;
k = (bx-ax)*(by-ay);
L = max({ax, ay/z});
U = min({bx, by/z});
return(log(((U *fabs(U))- (L *fabs(L)))/(2*k))) ;
}
real q_tilde_theta_lpdf(real z, real alpha, real mX, real MX, real mY, real MY){
return(alpha * uniform_lpdf(z |0, 5) + (1-alpha)*fZ_exact_lpdf(z | mX, MX, mY, MY));
}
}
data{
real<lower=0> a_alpha;
real<lower=0> b_alpha;
real<lower=0> max_X;
real<lower=0, upper=max_X> min_X;
real<lower=0> max_Y;
real<lower=0, upper=max_Y> min_Y;
}
parameters {
real<lower=0, upper=1> alpha;
real<lower=min_X, upper=max_X> X;
real<lower=min_Y, upper=max_Y> Y;
}
model{
alpha ~ beta(a_alpha, b_alpha);
target += q_tilde_theta_lpdf(Y/X | alpha, min_X, max_X, min_Y, max_Y) + uniform_lpdf( X | min_X, max_X)
}
generated quantities{
real<lower=min_Y/max_X, upper=max_Y/min_X> Z;
Z = Y/X;
}
'
######################
library(rstan)
```

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

```
## Loading required package: StanHeaders
```

```
## rstan (Version 2.17.3, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

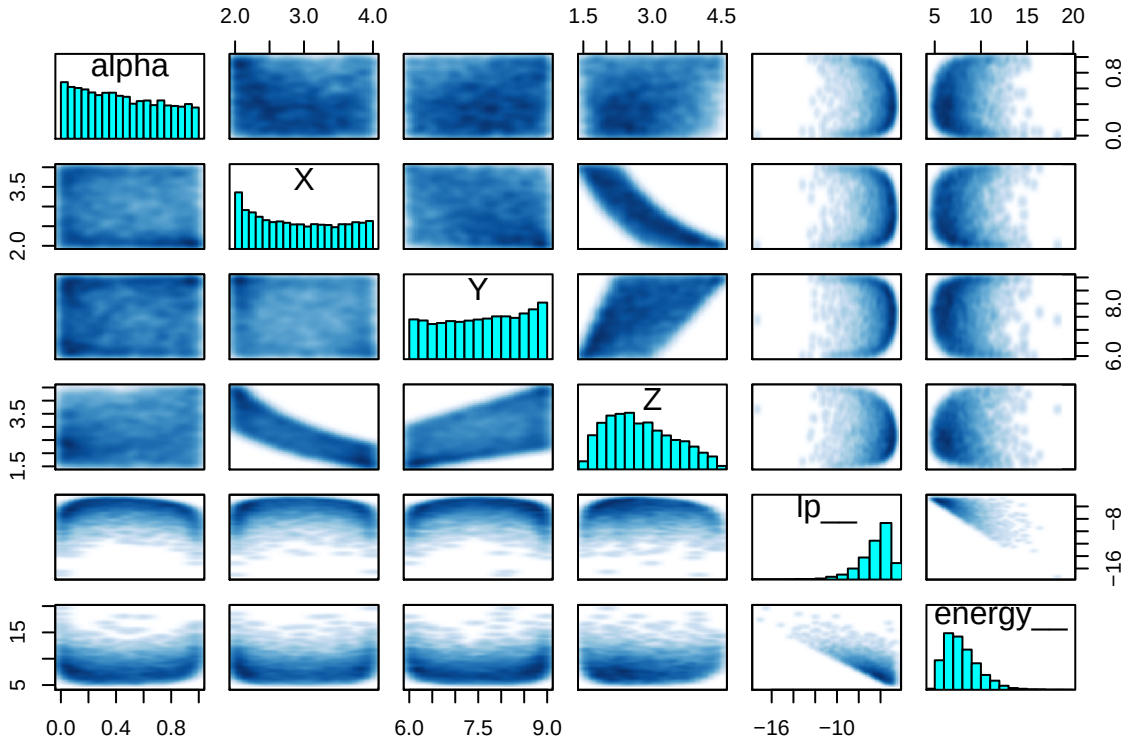
```
options(mc.cores = parallel::detectCores())

varying_alpha_run <- stan(model_code = varying_alpha,
                          data = list(a_alpha = 1,
                                      b_alpha = 1,
                                      min_X = 2, max_X = 4,
                                      min_Y = 6, max_Y = 9),
                          iter = 5000,
                          init = list(
                            chain1 = list(X = 3.2, Y = 7),
                            chain2 = list(X = 3.5, Y = 6.5),
                            chain3 = list(X = 3, Y = 7),
                            chain4 = list(X = 2.1, Y = 8)
                          )
)
```

```
## In file included from /home/max/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/config.hpp:39:0,
##                  from /home/max/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/math/tools/config
##                  from /home/max/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/
##                  from /home/max/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/
##                  from /home/max/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/
##                  from /home/max/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/
##                  from /home/max/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math.hpp:
##                  from /home/max/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/src/stan/model/
##                  from file754e42c017bf.cpp:8:
## /home/max/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/config/compiler/gcc.hpp:186:0: warning:
##  #  define BOOST_NO_CXX11_RVALUE_REFERENCES
##  ^
## <command-line>:0:0: note: this is the location of the previous definition
```

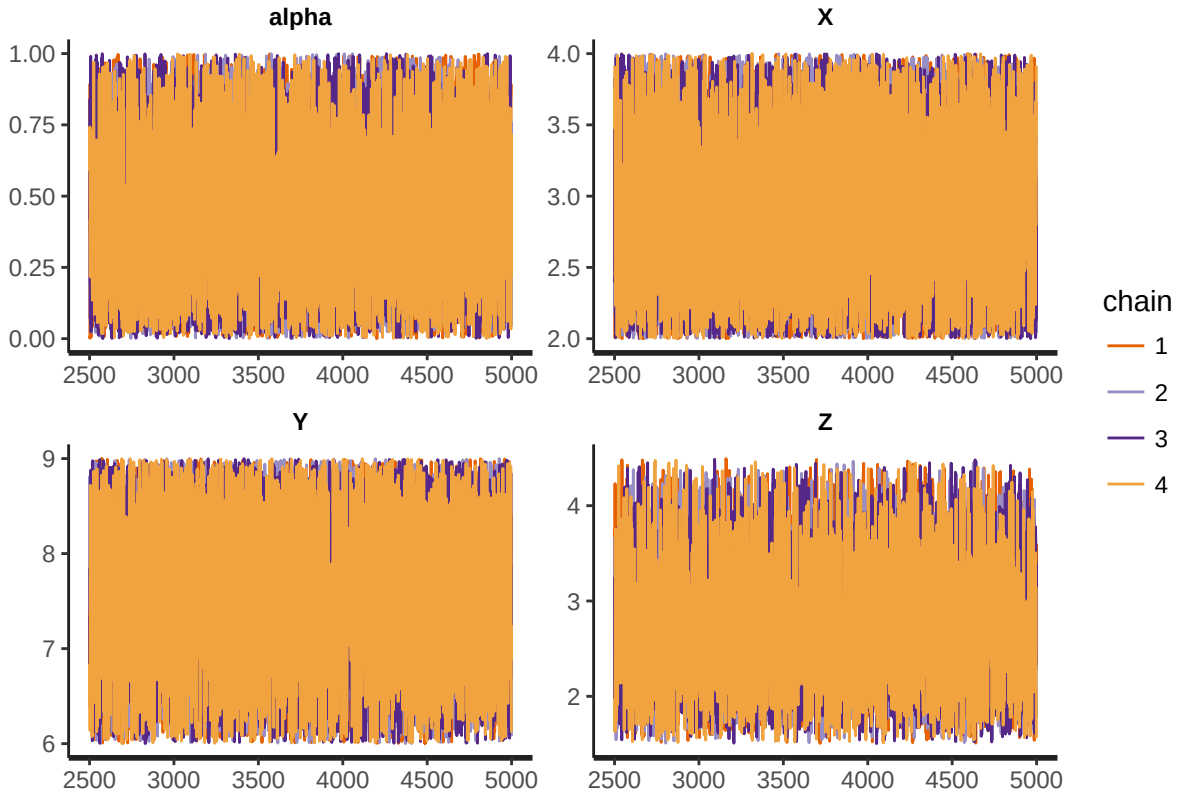
```
######################
varying_alpha_run
```

```
## Inference for Stan model: e853b54465aefea6906297a57199ace2.
## 4 chains, each with iter=5000; warmup=2500; thin=1;
## post-warmup draws per chain=2500, total post-warmup draws=10000.
##
##        mean se_mean   sd   2.5%   25%   50%   75% 97.5% n_eff Rhat
## alpha  0.45    0.00 0.29   0.02  0.20  0.43  0.69  0.97  8195    1
## X      2.90    0.01 0.61   2.02  2.33  2.85  3.44  3.95  7314    1
## Y      7.59    0.01 0.89   6.07  6.84  7.65  8.38  8.95  8090    1
## Z      2.75    0.01 0.72   1.64  2.17  2.66  3.28  4.25  6442    1
## lp__  -6.44    0.02 1.43 -10.16 -7.14 -6.09 -5.37 -4.76  4175    1
##
## Samples were drawn using NUTS(diag_e) at Thu Feb  8 18:50:19 2018.
## 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).
```

```
pairs(varying_alpha_run)
```



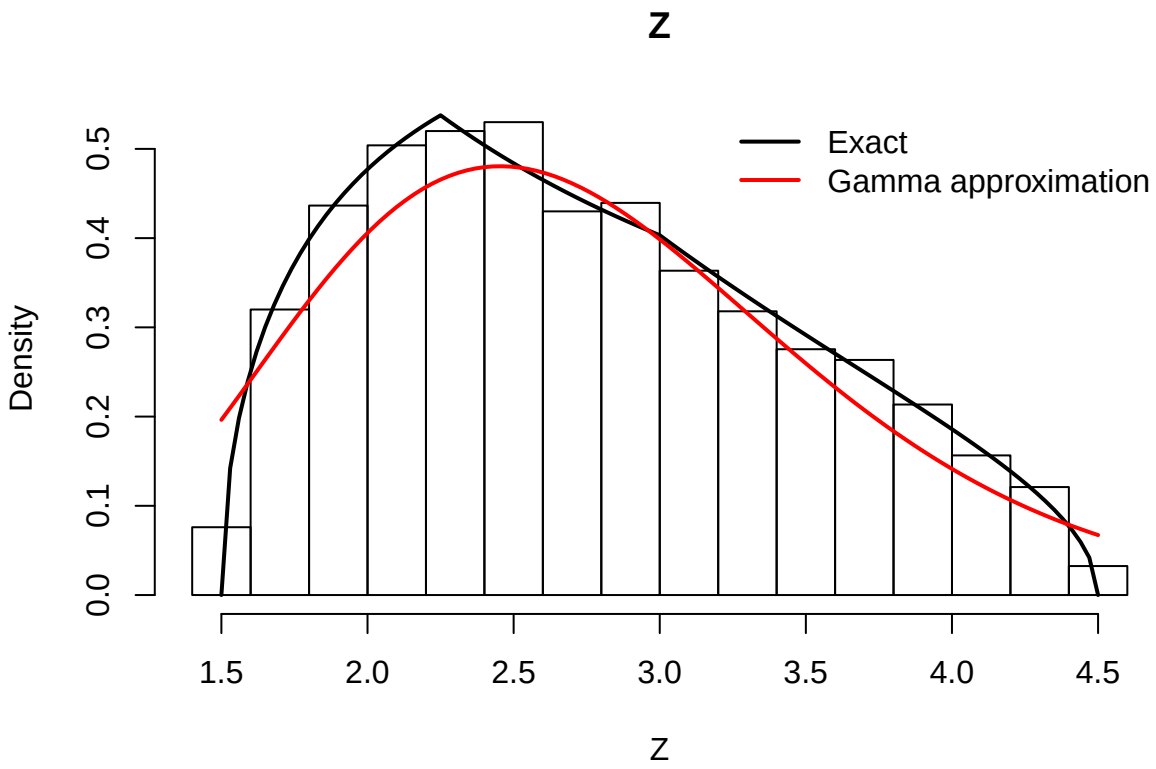

```
stan_trace(varying_alpha_run)
```



```r
source("../code/pooling_aux.r")
devtools::source_url("https://raw.githubusercontent.com/maxbiostat/CODE/b8473512151b0d205fd843bc291e45ea
```

```
## SHA-1 hash of file is a1dfa5d771fdeb74b331d3314628574167 46eb31
```

```r
dZ_exact <- function(x) dpoolnorm.positive(x = x, D = list(function(x) {dunif(x, 0, 5)},
                                                           function(x) {analytic_Z(x, ax = 2, bx = 4, a
                                           alphas = c(.5, .5)
)
dZ_approx <- function(x) dpoolnorm.positive(x = x, D = list(function(x) {dunif(x, 0, 5)},
                                                            function(x) {dgamma(x, 18.3, 7.05)}),
                                            alphas = c(.5, .5)
)

Z_samples <- extract(varying_alpha_run, 'Z')$Z
hist(Z_samples,
     probability = TRUE, main = "Z", xlab = expression(Z))
curve(dZ_exact, 1.5, 4.5, lwd = 2, add = TRUE)
curve(dZ_approx, 1.5, 4.5, lwd = 2, col = 2, add = TRUE)
legend(x = "topright", legend = c("Exact", "Gamma approximation"), col = 1:2, lwd = 2, bty = "n")
```




```r
mu <- integrate( function(x) x * dZ_exact(x), 0 , Inf)
sq <- integrate( function(x) x^2 * dZ_exact(x), 0 , Inf)

mean(Z_samples); mu$value
```

```
## [1] 2.753282
```

```
## [1] 2.744048
```

```r
var(Z_samples); sq$value-mu$value^2
```

```
## [1] 0.5243601
```

```
## [1] 0.5224814
```

```r
Alpha_samples <- extract(varying_alpha_run, 'alpha')$alpha
hist(Alpha_samples,
     probability = TRUE, main = "Alpha", xlab = expression(alpha))
curve(dbeta(x, 1, 1), lwd = 2, add = TRUE)
```