Truncated Multivariate Normal Variates in Stan

Ben Goodrich

May 1, 2017

1 Introduction

There are many situations in which we would like to draw from a truncated multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and variance-covariance matrix $\boldsymbol{\Sigma}$. Or we would like place a truncated multivariate normal distribution on some parameter or outcome of a data-generating process.

Historically, the usual approach is to use a Gibbs sampler to draw from the K full-conditional distributions, which are each univariate truncated normal. If we let $-k$ indicate "all but the k-th", denote the lower bound (which is possibly $-\infty$) by $\underline{b_k}$, and denote the upper bound (which is possibly $\overline{b_k}$), the truncated normal full-conditional distribution of y_k is

$$y_k|\,\mathbf{y}_{-k},\boldsymbol{\mu},\boldsymbol{\Sigma},\underline{b_k},\overline{b_k} \quad \sim \quad \mathcal{TN}_{\underline{b_k}\leq Y_k\leq\overline{b_k}}\left(\mu_k+\boldsymbol{\Sigma}_{k,-k}\boldsymbol{\Sigma}_{-k,-k}^{-1}\left(\mathbf{y}_{-k}-\boldsymbol{\mu}_{-k}\right),\Sigma_{k,k}-\boldsymbol{\Sigma}_{k,-k}\boldsymbol{\Sigma}_{-k,-k}^{-1}\boldsymbol{\Sigma}_{k,-k}^{\top}\right)$$

As with any Gibbs sampler, the convergence can be slow when the conditional variance is near zero.

In any event, Stan employs a variant of Hamiltonian Monte Carlo rather than Gibbs sampling, so it would be useful to have a way to use a truncated multivariate normal distribution in Stan. Whereas Gibbs samplers use *full*-conditional distributions, Stan uses the joint kernel, which can be written telescopically as a marginal density of the first variate times a product of *partial*-conditional distributions given all previous random variates. In other words, if $k'=k-1$, we could write

$$\begin{aligned}
y_1 \quad &\sim \quad \mathcal{TN}_{\underline{b_1}\leq Y_1\leq\overline{b_1}}\left(\mu_1,\Sigma_{1,1}\right)\\
y_2|\,y_1 \quad &\sim \quad \mathcal{TN}_{\underline{b_2}\leq Y_2\leq\overline{b_2}}\left(\mu_2+\Sigma_{2,1}\Sigma_{1,1}^{-1}\left(y_1-\mu_1\right),\Sigma_{2,2}-\Sigma_{2,1}\Sigma_{1,1}^{-1}\Sigma_{2,1}\right)\\
&\vdots\\
y_k|\,\mathbf{y}_{1:k'} \quad &\sim \quad \mathcal{TN}_{\underline{b_k}\leq Y_k\leq\overline{b_k}}\left(\mu_k+\Sigma_{k,1:k'}\Sigma_{1:k',1:k'}^{-1}\left(\mathbf{y}_{1:k'}-\boldsymbol{\mu}_{1:k'}\right)\Sigma_{k,k}-\Sigma_{k,1:k'}\Sigma_{1:k',1:k'}^{-1}\Sigma_{k,1:k'}^{\top}\right)
\end{aligned}$$

However, this way of writing a joint kernel is not that useful because we need a way to ensure that $\underline{b_k}\leq y_k\leq\overline{b_k}$ and if we had such a construction, then we would not need to evaluate a truncated normal density. The rest of this paper establishes the necessary construction using the Cholesky factor of $\boldsymbol{\Sigma}$.

2 Multivariate Transformations

This section reviews the stochastic representation of the multivariate normal distribution, with an emphasis on its Cholesky factor. If the K-vector $\mathbf{y}$ is distributed multivariate normal with mean vector $\boldsymbol{\mu}$ and positive-definite variance-covariance matrix $\boldsymbol{\Sigma}$, then we can write

$$\mathbf{y} \quad \overset{d}{=} \quad \boldsymbol{\mu}+\mathbf{L}\mathbf{z},$$

where z_k is independently and identically distributed univariate standard normal and $\mathbf{L}$ is the Cholesky factor of $\boldsymbol{\Sigma}=\mathbf{L}\mathbf{L}^{\top}$.

This well-known result is worth proving. If $\mathbf{y}(\mathbf{z}) = \boldsymbol{\mu} + \mathbf{L}\mathbf{z}$, then the inverse transformation is $\mathbf{z}(\mathbf{y}) = \mathbf{L}^{-1}(\mathbf{y} - \boldsymbol{\mu})$, so the Jacobian matrix of the transformation from $\mathbf{y}$ to $\mathbf{z}$ is $\mathbf{J} = \mathbf{L}^{-1}$ and its determinant is $|\mathbf{J}| = \left|\mathbf{L}^{-1}\right| = \frac{1}{|\mathbf{L}|} = \frac{1}{\prod_{k=1}^{K} L_{kk}} > 0$. Since each z_k is independently and identically distributed univariate standard normal, $\mathbf{z}$ is distributed multivariate normal with mean vector $\mathbf{0}$ and variance-covariance matrix $\mathbf{I}$. If we substitute $\mathbf{L}^{-1}(\mathbf{y} - \boldsymbol{\mu})$ for $\mathbf{z}$ in this multivariate normal density and account for the change in hypervolume, we get

$$\begin{aligned} f_Y(\mathbf{y}|\,\boldsymbol{\mu}, \mathbf{L}) &= \frac{1}{|\mathbf{L}|} \times f_Z(\mathbf{z}(\mathbf{y})) = \frac{1}{|\mathbf{L}|} \times \frac{1}{\prod_{k=1}^{K}\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\mathbf{L}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right)^{\top}\left(\mathbf{L}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right)\right) \\ &= \frac{1}{|\mathbf{L}|(2\pi)^{\frac{K}{2}}} \exp\left(-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^{\top}\mathbf{L}^{-\top}\mathbf{L}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right). \end{aligned}$$

However, probability theorists who are not burdened by the requirement of having to actually estimate anything on a finite-precision computer tend to parameterize the multivariate normal distribution in terms of $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^{\top}$. Making the substitutions that $\boldsymbol{\Sigma}^{-1} = \mathbf{L}^{-\top}\mathbf{L}^{-1}$ and $|\mathbf{L}| = |\boldsymbol{\Sigma}|^{\frac{1}{2}}$, we can obtain the conventional parameterization of the multivariate normal density $f_Y(\mathbf{y}|\,\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{|\boldsymbol{\Sigma}|^{\frac{1}{2}}(2\pi)^{\frac{K}{2}}} \exp\left(-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right)$.

The parameterization of the multivariate normal density in terms of the Cholesky factor is not only preferable numerically but is also convenient for truncation. We can partition the Cholesky factor as

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{L}_{k1} & L_{kk} & \mathbf{0} \\ \mathbf{L}_{31} & \mathbf{L}_{3k} & \mathbf{L}_{33} \end{bmatrix},$$

where $\mathbf{L}_{11}$ and $\mathbf{L}_{33}$ are lower-triangular, $\mathbf{L}_{31}$ is generally a dense submatrix, $L_{kk} > 0$ is a scalar, $\mathbf{L}_{k1}$ is a row-vector consisting of the $k-1$ elements of $\mathbf{L}$ to the left of L_{kk}, and $\mathbf{L}_{3k}$ is a column-vector consisting of the $k-1$ elements below L_{kk}. Similarly, we can partition $\mathbf{y}$, $\boldsymbol{\mu}$, and $\mathbf{z}$ conformably as

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ y_k \\ \mathbf{y}_3 \end{bmatrix} \quad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \mu_k \\ \boldsymbol{\mu}_3 \end{bmatrix} \quad \mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ z_k \\ \mathbf{z}_3 \end{bmatrix}.$$

Thus, we can write

$$\begin{aligned} \begin{bmatrix} \mathbf{y}_1 \\ y_k \\ \mathbf{y}_3 \end{bmatrix} &\overset{d}{=} \begin{bmatrix} \boldsymbol{\mu}_1 \\ \mu_k \\ \boldsymbol{\mu}_3 \end{bmatrix} + \begin{bmatrix} \mathbf{L}_{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{L}_{k1} & L_{kk} & \mathbf{0} \\ \mathbf{L}_{31} & \mathbf{L}_{3k} & \mathbf{L}_{33} \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ z_k \\ \mathbf{z}_3 \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{\mu}_1 \\ \mu_k \\ \boldsymbol{\mu}_3 \end{bmatrix} + \begin{bmatrix} \mathbf{L}_{11}\mathbf{z}_1 & + & 0 & + & 0 \\ \mathbf{L}_{k1}\mathbf{z}_1 & + & L_{kk}z_k & + & 0 \\ \mathbf{L}_{31}\mathbf{z}_1 & + & \mathbf{L}_{3k}z_k & + & \mathbf{L}_{33}\mathbf{z}_3 \end{bmatrix}. \end{aligned}$$

3 Truncated Multivariate Normal

The previous section is sufficient if there are no constraints on $\mathbf{y}$. If there are constraints on $\mathbf{y}$, we can express them as constraints on $\mathbf{z}$. In some situations, the mean vector will be a function of parameters, such as $\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\beta}$, but that does not affect the following scheme to draw from a truncated multivariate normal distribution.

Let $z(u) = \Phi^{-1}(u)$, where u is distributed standard uniform and $\Phi^{-1}(\cdot)$ is the inverse CDF of the standard normal distribution. In other words, $z(u)$ could be generated by the inverse CDF method, and we can write

$$\mathbf{y} \overset{d}{=} \boldsymbol{\mu} + \mathbf{L}\mathbf{z}(\boldsymbol{u}),$$

where $\mathbf{u}$ is a vector of standard uniform variates.

Suppose there is a known bound, b_1, on $y_1 = \mu_1 + L_{11} z(u_1)$. We can solve for $\frac{b_1 - \mu_1}{L_{11}} = z^*(u_1)$ — so that the constraint binds if $b_1 = z^*(u_1)$ — and then solve for $u_1^* = \Phi\left(\frac{b_1-\mu_1}{L_{11}}\right)$, where $\Phi(\cdot)$ is the CDF of the standard normal distribution. The constraint on y_1 eliminates part of the support for the uniform variate. If $b_1 = \overline{b_1}$ is an upper bound on y_1, then $v_1 = u_1 u_1^*$ is uniform between 0 and u_1^* with density $\frac{1}{u_1^*}$. If $b_1 = \underline{b_1}$ is a lower bound on y_1, then $v_1 = u_1^* + (1-u_1^*) u_1$ is uniform between u_1^* and 1 with density $\frac{1}{1-u_1^*}$.

Given a realization of u_1 and thus $z_1 = \Phi^{-1}(u_1)$, we can consider a known bound, b_2, on $y_2 = \mu_2 + L_{21} z_1 + L_{22} z(u_2)$. We can solve for $\frac{y_2 - (\mu_2 + L_{21} z_1)}{L_{22}}$ — so that the constraint binds if $b_2 = z^*(u_2)$ — and then solve for $u_2^* = \Phi\left(\frac{y_2 - (\mu_2 + L_{21} z_1)}{L_{22}}\right)$. The constraint on y_2 eliminates part of the support for the uniform variate. If $b_2 = \overline{b_2}$ is an upper bound on y_2, then $v_2 = u_2 u_2^*$ is uniform between 0 and u_2^* with density $\frac{1}{u_2^*}$. If $b_2 = \underline{b_2}$ is a lower bound on y_2, then $v_2 = u_2^* + (1-u_2^*) u_2$ is uniform between u_2^* and 1 with density $\frac{1}{1-u_2^*}$.

In general, given realizations of $u_1 \cdots u_{k-1}$ and thus $\mathbf{z}_1 = \Phi^{-1}\left(\begin{bmatrix} u_1 & \cdots & u_{k-1}\end{bmatrix}\right)$, we can consider a known bound, b_k, on $y_k = \mu_k + \mathbf{L}_{k1}\mathbf{z}_1 + L_{kk} z_k$. We can solve for $\frac{y_k - (\mu_k + \mathbf{L}_{k1}\mathbf{z}_1)}{L_{kk}}$ — so that the constraint binds if $b_k = z^*(u_k)$ — and then solve for $u_k^* = \Phi\left(\frac{y_k - (\mu_k + \mathbf{L}_{k1}\mathbf{z}_1)}{L_{kk}}\right)$. The constraint on y_k eliminates part of the support for the uniform variate. If $b_k = \overline{b_k}$ is an upper bound on y_k, then $v_k = u_k u_k^*$ is uniform between 0 and u_k^* with density $\frac{1}{u_k^*}$. If $b_k = \underline{b_k}$ is a lower bound on y_k, then $v_k = u_k^* + (1-u_k^*) u_k$ is uniform between u_k^* and 1 with density $\frac{1}{1-u_k^*}$.

If there happens to be no constraint on $y_k = \mu_k + \mathbf{L}_{k1}\mathbf{z}_1 + L_{kk} z_k$, then we simply set $z_k = \Phi^{-1}(u_k)$, where u_k is standard uniform and thus has density of 1. In rare cases, there may be both a lower bound and an upper bound on y_k, in which case we can combine the previous results such that $v_k = \underline{u_k^*} + \left(\overline{u_k^*} - \underline{u_k^*}\right) u_k$ is uniform between the implied lower bound, $\underline{u_k^*}$, and the implied upper bound, $\overline{u_k^*}$ with density $\frac{1}{\overline{u_k^*} - \underline{u_k^*}}$.

4 Implementation in a Stan Program

Algorithm 1 contains a complete Stan program to draw from a truncated multivariate normal distribution. First, we define a function called `make_stuff` that constructs $\mathbf{z}(\mathbf{u})$ and a vector of K derivatives that is explained below. We could call the `expose_stan_functions` function to expose `make_stuff` to R and verify that it is working correctly.

Second, we pass K and $\mathbf{b}$ to the `data` block of the Stan program in addition to a K-vector $\mathbf{s}$ whose typical element is

$$s_k = \begin{cases} -1 & \text{if } b_k \text{ is an upper bound} \\ 0 & \text{if } y_k \text{ is unconstrained} \\ 1 & \text{if } b_k \text{ is a lower bound} \end{cases}.$$

Also, we pass $\boldsymbol{\mu}$ and $\mathbf{L}$, although it would be straightforward to declare these as parameters and estimate them if we had other data.

In the `parameters` block, we would prefer to declare $\mathbf{v}$ as a K-vector that is uniform on some subset of the unit interval. However, Stan only permits scalar bounds on vectors declared in the parameters block. Thus, we instead have to declare $\mathbf{u}$ as a K-vector that is uniform on the unit interval and construct each element of $\mathbf{v}$ as an intermediate. We essentially have a “prior” on v_k conditional on b_k, s_k, $\boldsymbol{\mu}_{1:k}$, $\mathbf{L}_{1:k,1:k}$, and $\mathbf{z}(\mathbf{u}_{1:k'})$ whose density is

$$f_V(v_k | b_k, s_k, u_k^*) = \begin{cases} \frac{1}{u_k^*} & \text{if } b_k \text{ is an upper bound} \\ \frac{1}{1-u_k^*} & \text{if } b_k \text{ is a lower bound} \end{cases}.$$

Hence, in the `model` block, we have to adjust the log-kernel by the logarithm of the derivative of the transformation from v_k to u_k, which is

$$\ln \frac{\partial}{\partial v_k} u_k \sim \begin{cases} \ln u_k^* & \text{if } b_k \text{ is an upper bound} \\ \ln(1-u_k^*) & \text{if } b_k \text{ is a lower bound} \end{cases}.$$

Algorithm 1 Stan Program to Draw from a Truncated Multivariate Normal

```
functions {
  vector[] make_stuff(vector mu, matrix L, vector b, vector s, vector u) {
    int K = rows(mu); vector[K] d; vector[K] z; vector[K] out[2];
    for (k in 1:K) {
      int km1 = k - 1;
      if (s[k] != 0) {
        real z_star = (b[k] -
                      (mu[k] + ((k > 1) ? L[k,1:km1] * head(z, km1) : 0))) /
                      L[k,k];
        real v; real u_star = Phi(z_star);
        if (s[k] == -1) {
          v = u_star * u[k];
          d[k] = u_star;
        }
        else {
          d[k] = 1 - u_star;
          v = u_star + d[k] * u[k];
        }
        z[k] = inv_Phi(v);
      }
      else {
        z[k] = inv_Phi(u[k]);
        d[k] = 1;
      }
    }
    out[1] = z;
    out[2] = d;
    return out;
  }
}
data {
  int<lower=2> K;             // number of dimensions
  vector[K] b;                // lower or upper bound

  // s[k] ==  0 implies no constraint; otherwise
  // s[k] == -1 -> b[k] is an upper bound
  // s[k] == +1 -> b[k] is a lower bound
  vector<lower=-1,upper=1>[K] s;

  vector[K] mu;
  cholesky_factor_cov[K,K] L;
}
parameters {
  vector<lower=0,upper=1>[K] u;
}
model {
  target += log(make_stuff(mu, L, b, s, u)[2]); // Jacobian adjustments
  // implicit: u ~ uniform(0,1)
}
generated quantities {
  vector[K] y = mu + L * make_stuff(mu, L, b, s, u)[1];
}
```

These derivatives are returned as the second vector in `make_stuff`. Finally, in the `generated quantities` block, we construct the K-vector $\mathbf{y} = \boldsymbol{\mu} + \mathbf{Lz}(\mathbf{u})$, where is $\mathbf{z}(\mathbf{u})$ is the first vector returned by `make_stuff`.

To call this Stan program from R, we can execute

```
K <- 2
rho <- 0.5
Sigma <- matrix(c(1,rho,rho,1), K, K)
standata <- list(K = K, b = c(1/pi, exp(-1)), s = c(1,-1), mu = c(0,0),
                 L = t(chol(Sigma)))
library(rstan)
rstan_options(auto_write = TRUE)
post <- stan("tMVN.stan", data = standata)
```

which results in

```
print(post, digits = 3)

## Inference for Stan model: tMVN.
## 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
## u[1]   0.433   0.004 0.270  0.020  0.201  0.413  0.647  0.937  3845 1.000
## u[2]   0.497   0.005 0.288  0.024  0.250  0.489  0.747  0.978  3392 0.999
## y[1]   0.876   0.008 0.448  0.339  0.525  0.772  1.115  1.983  3411 1.000
## y[2]  -0.307   0.009 0.514 -1.549 -0.606 -0.201  0.098  0.344  3012 0.999
## lp__  -5.702   0.030 1.199 -8.955 -6.172 -5.325 -4.831 -4.524  1551 1.002
##
## Samples were drawn using NUTS(diag_e) at Mon May  1 03:17:57 2017.
## 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).
```

It is evident from Figure 1 that the posterior distribution of u_1 is not uniform due to the constraints on y_1 and y_2 that make large values of u_1 unlikely. Nevertheless, the posterior distribution is not difficult for Stan to sample efficiently from, although it could become more difficult as K increases and as the bounds eliminate more of the unconstrained density.

We can compare the results to those obtained via rejection sampling with

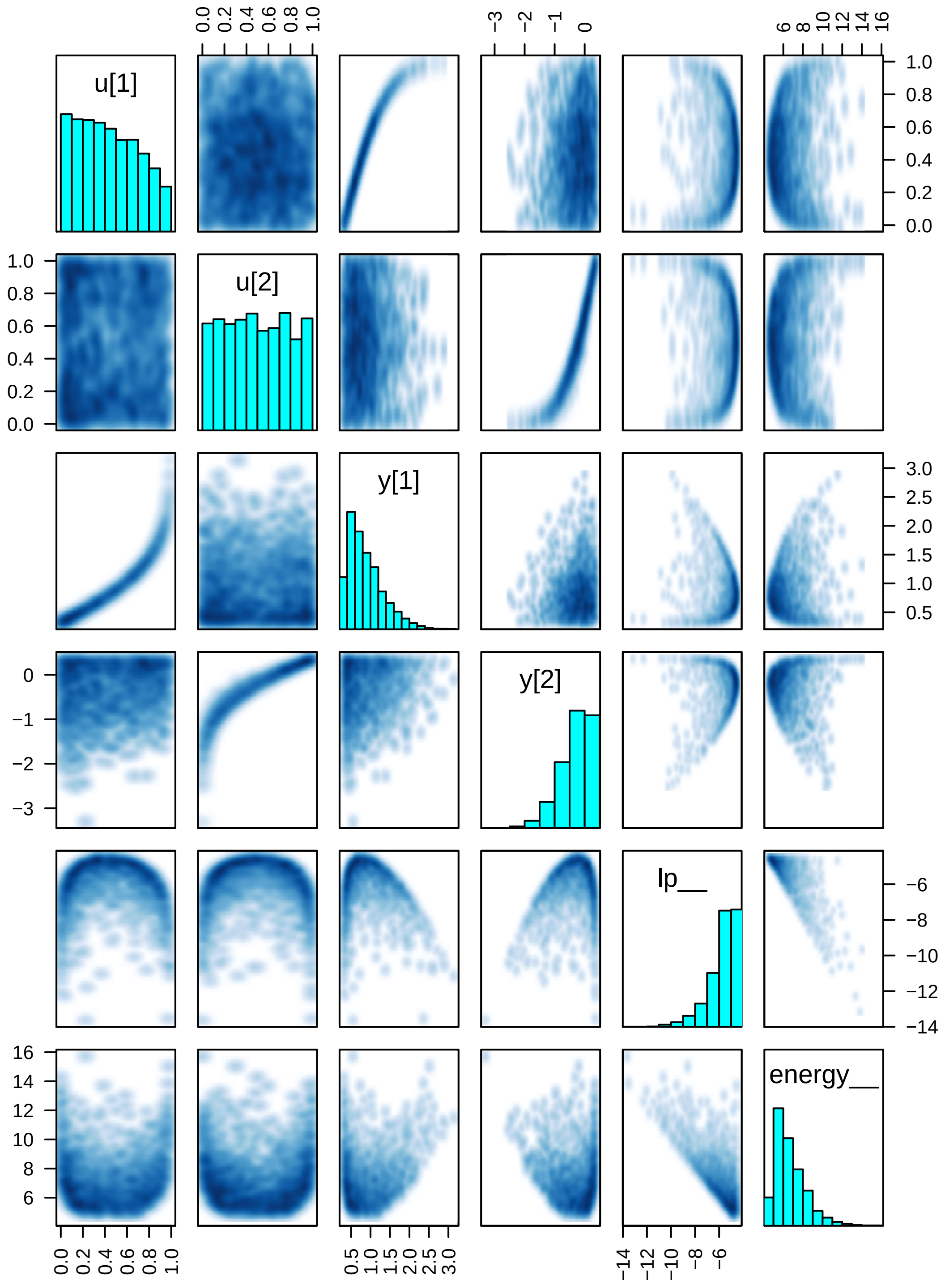
```
library(mvtnorm)
y_raw <- rmvnorm(16000, sigma = Sigma)
y <- y_raw[y_raw[,1] > (1 / pi) & y_raw[,2] < exp(-1), ]
round(digits = 3, t(apply(y, 2, FUN = function(x) {
  c(mean = mean(x), sd = sd(x),
    quantile(x, probs = c(0.025, 0.25, 0.5, 0.75, 0.975)))
})))

##         mean    sd   2.5%    25%    50%   75% 97.5%
## [1,]   0.885 0.454  0.339  0.527  0.776 1.138 1.984
## [2,]  -0.298 0.514 -1.537 -0.592 -0.189 0.107 0.345
```

Stan is, in a manner of speaking, more efficient than rejection sampling in that it obtains an effective sample size of a few thousand from a nominal sample size of 4000 (after thowing away 4000 realizations as warmup). To obtain a

Figure 1: Pairs Plot for a Truncated Bivariate Normal

```r
pairs(post, las = 2)
```



similar number of independent realizations via rejection sampling, you have to take an unconstrained sample of about $16,000$.

```
c(y_raw = nrow(y_raw), y = nrow(y))

## y_raw     y
## 16000  2708
```