A Simple Demo on Caching R Objects and Graphics with **pgfSweave**

Yihui Xie

January 2, 2011

Caching is often necessary in our daily statistical computation. Fortunately the R packages **cacheSweave** and **pgfSweave** have provided functionalities to cache R objects and graphics respectively. This short article contains a simple demo using Gibbs sampling to generate random numbers from a bivariate Normal distribution. We need to introduce a little bit about the background first.

For the bivariate Normal distribution

$$\left[\begin{array}{c} X \\ Y \end{array}\right] \sim \mathcal{N}\left(\left[\begin{array}{c} \mu_X \\ \mu_Y \end{array}\right], \left[\begin{array}{cc} \sigma_X^2 & \rho\sigma_X\sigma_Y \\ \rho\sigma_X\sigma_Y & \sigma_Y^2 \end{array}\right]\right)$$

we know the conditional distributions

$$\begin{aligned} Y|X = x &\sim \mathcal{N}\left(\mu_Y + \frac{\sigma_Y}{\sigma_X}\rho(x-\mu_X),\, (1-\rho^2)\sigma_Y^2\right) \\ X|Y = y &\sim \mathcal{N}\left(\mu_X + \frac{\sigma_X}{\sigma_Y}\rho(y-\mu_Y),\, (1-\rho^2)\sigma_X^2\right) \end{aligned}$$

so we can use the Gibbs sampling to generate random numbers from the joint Normal distribution. First initialize $x^{(0)}$ and $y^{(0)}$, then repeatedly generate $x^{(k)} \sim f(x|y^{(k-1)})$ and $y^{(k)} \sim f(y|x^{(k)})$ (these two conditional distributions are given above).

The Gibbs sample algorithm is fairly easy to implement:

```
## this Gibbs sampling algorithm can be time-consuming
## when n is large
rbinormal = function(n, mu1, mu2, sigma1, sigma2, rho) {
    x = rnorm(1, mu1, sigma1)
    y = rnorm(1, mu2, sigma2)
    xy = matrix(nrow = n, ncol = 2)
    for (i in 1:n) {
        x = rnorm(1, mu1 + sigma1/sigma2 * rho * (y - mu2), sqrt(1 - rho^2) *
            sigma1)
        y = rnorm(1, mu2 + sigma2/sigma1 * rho * (x - mu1), sqrt(1 - rho^2) *
            sigma2)
        xy[i, ] = c(x, y)
    }
    xy
}
## this may take more than 30 seconds
dat = rbinormal(5e+05, 0, 1, 2, 3, 0.7)
```

Note this code chunk is cached, so next we can use the object `dat` directly, e.g. the marginal means and standard deviations as well as the covariance matrix are as follows:

```
## marginal means and standard deviations of the sample
apply(dat, 2, mean)
```

```r
library(KernSmooth)
est = bkde2D(dat, c(0.2, 0.2))
par(mar = c(4, 4, 1.5, 0.1), mgp = c(2, 0.9, 0))
plot(dat[sample(nrow(dat), 500), ], pch = 20, cex = 0.5, xlab = "$X$", ylab = "$Y$",
    col = "red", main = "$(X,Y) \\sim MVN(\\mu,\\Sigma)$", cex.main = 1)
abline(v = 0, h = 1, lty = 2, col = "darkgray")
contour(est$x1, est$x2, est$fhat, nlevels = 15, drawlabels = FALSE, add = TRUE,
    col = "darkgray")
```

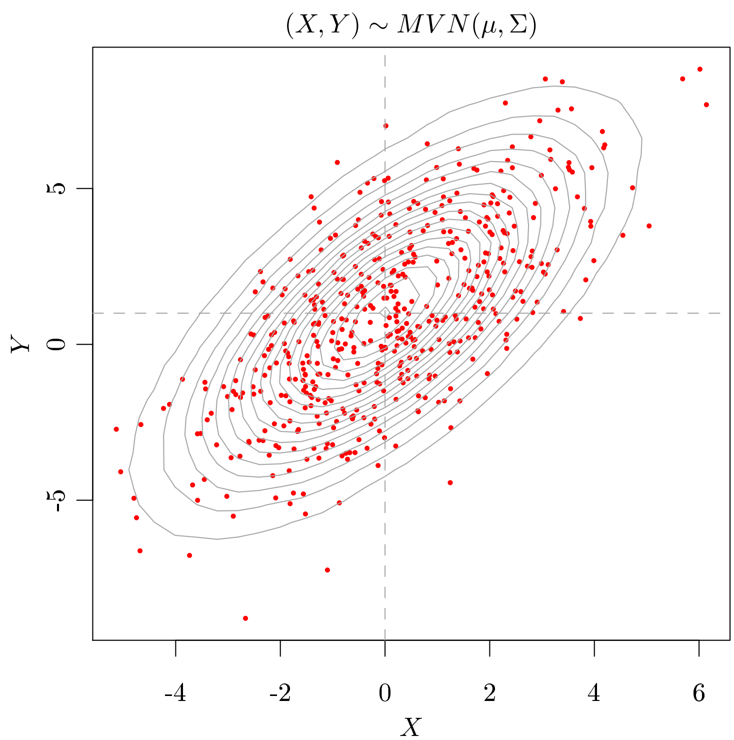

Figure 1: Distribution of the simulated random numbers from the bivariate Normal distribution with $\mu_X = 0,\ \sigma_X = 2,\ \mu_Y = 1,\ \sigma_Y = 3,\ \rho = 0.7$.

```
[1] 0.0034406 1.0023501
```

```r
apply(dat, 2, sd)
```

```
[1] 2.003180 3.005288
```

```r
cov(dat)
```

```
         [,1]     [,2]
[1,] 4.012730 4.216852
[2,] 4.216852 9.031754
```

Figure 1 shows the bivariate distribution with a contour plot, which we can expect from the structure of the bivariate Normal distribution. Note that only 500 sample points are shown on the plot.