Bayesian Inference and Computational Methods 2009

Homework 3 – Solutions

1. Some useful **R** commands are:

```
z = runif(10000)                # generate sample
hist(z)                         # simple check for uniformity
plot(sort(z), ty='l')           # better check for uniformity
plot(z[-10000],z[-1],pch=3)     # check for first-order independence
```

2. (a) Possible **R** commands for $Z_1 = \max(Y_1, Y_2)$ are:

```
y1 = runif(10000)                             # first sample
y2 = runif(10000)                             # second sample
z1 = pmax(y1,y2)                              # successive maxima
hist(z1)                     # compare with theoretical density 2x
plot(ppoints(10000)^0.5,sort(z1),pch=3)                 # Q-Q plot
mean(z1)                     # compare with theoretical mean 2/3
```

   Possible **R** commands for $Z_2 = \min(Y_1, Y_2)$ are:

```
y1 = runif(10000)                             # first sample
y2 = runif(10000)                             # second sample
z2 = pmin(y1,y2)                              # successive minima
hist(z2)                     # compare with theoretical density 2(1-x)
plot(1-(1-ppoints(10000))^0.5,sort(z2),pch=3)      # Q-Q plot
mean(z2)                     # compare with theoretical mean 1/3
```

   (b) This is similar to (a). Note that in this case there is no easy probability plot for verifying the distribution of the *maximum*. However, recall that if $Y_i \sim \text{Exp}(\lambda_i)$ are independent random variables then $\min_i Y_i \sim \text{Exp}(\sum_i \lambda_i)$. Appropriate **R** code to construct a probability plot to check whether data `z` may reasonably be modelled as a random sample from an $\text{Exp}(\lambda)$ distribution is given by

```
plot(-log(1-ppoints(length(z))),sort(z),pch=3)
```

   the plot then corresponding to an approximate straight line of slope λ^{-1} through the origin.

   (c) This is again similar to (a). There are no easy probability plots.

3. For the given logistic distribution we have $F^{-1}(p) = -\log(p^{-1} - 1)$. Hence suitable **R** code to generate a random sample of size 10000 from this distribution is

```
y = -log(1/runif(10000) - 1)
```

   The true mean and variance are 0 (check the symmetry of the density function) and $\pi^2/3$ respectively.

4. Suitable **R** code for simulating the total shown by three dice, and displaying and tabulating the probability function is

```
u = runif(10000)
sample1 = 1 + (u>1/6) + (u>2/6) + (u>3/6) + (u>4/6) + (u>5/6)
u = runif(10000)
sample2 = 1 + (u>1/6) + (u>2/6) + (u>3/6) + (u>4/6) + (u>5/6)
u = runif(10000)
sample3 = 1 + (u>1/6) + (u>2/6) + (u>3/6) + (u>4/6) + (u>5/6)
dice.total = sample1 + sample2 + sample3
```

```
library(MASS)
truehist(dice.total)
table(dice.total)/10000
```

5. (a) The corresponding distribution function F on $[0,1]$ is given by $F(y) = (e^y - 1)/(e-1)$, $y \in [0,1]$, and hence $F^{-1}(p) = \log(1+(e-1)p)$, $p \in [0,1]$. Hence appropriate **R** code for simulation by the inverse transform method is

```
sample = log(1+(exp(1)-1)*runif(10000))
```

Appropriate code for simulation by rejection sampling using a $U \sim \mathrm{U}(0,1)$ envelope has already been given. The theoretical unconditional acceptance probability is then $e^{-1}\mathbf{E}e^U = 1 - e^{-1}$, which should correspond to the fraction of the sample from the envelope actually accepted.

(b) The corresponding distribution function F on $[0,1]$ is given by $F(y) = 1-(1-y)^3$, $y \in [0,1]$, and hence $F^{-1}(p) = 1-(1-p)^{1/3}$, $p \in [0,1]$. Hence appropriate **R** code for simulation by the inverse transform method is

```
sample = 1-(1-runif(10000))^(1/3)
```

Appropriate code for simulation by rejection sampling using a $U \sim \mathrm{U}(0,1)$ is

```
u = runif(10000)
acc = runif(10000) < (1-u)^2          #acceptance decisions
sample = u[acc]                       #sample with reqd dist
```

Since, conditional on generating u from the *envelope* distribution, the acceptance probability is $(1-u)^2$, it follows that the theoretical unconditional acceptance probability is $\mathbf{E}(1-U)^2 = 1/3$. This should again correspond to the fraction of the sample from the envelope actually accepted.

6. The difficulty of using the *inverse transform* method is that of inversion of the distribution function.

To simulate from the $\Gamma(2,1)$ distribution by *rejection sampling* with *envelope* the $\mathrm{Exp}(1/2)$ distribution, note that the latter distribution has density proportional to g where $g(y) = e^{-y/2}$. Since

$$\sup_{y\geq 0} \frac{f(y)}{g(y)} = \sup_{y\geq 0} ye^{-y/2} = 2e^{-1},$$

we may simulate from the $\mathrm{Exp}(1/2)$ distribution and *accept* each realisation y with probability $ye^{1-y/2}/2$. Appropriate **R** code is

```
y = rexp(10000,0.5)
acc = runif(10000) < 0.5*y*exp(1-y/2)
sample = y[acc]
```

To generate a further sample by using instead the result that a gamma random variable with integer shape parameter can be regarded as a sum of independent exponential random variables, we may use

```
y1 = rexp(10000)
y2 = rexp(10000)
sample = y1 + y2
```

In either case we may check that our sample does indeed come from a $\Gamma(2,1)$ distribution by using a Q-Q plot:

```
plot(qgamma(ppoints(sample),2,1), sort(sample), pch=3)
```

7. Appropriate **R** code to generate the sample and display summary statistics, histogram and exponential Q-Q plot is

```
y = rexp(10000)
ind = runif(10000) < 3/4
sample = ind*y + (!ind)*10*y
summary(sample)
mean(sample)
sqrt(var(sample))
hist(sample,n=80)
plot(-log(1-ppoints(10000)),sort(sample),pch=3)
```

   Note the very long tail of the distribution—and also that the distribution of the sample is *not* itself exponential (being instead a mixture of two exponential distributions).

8. Appropriate **R** code to generate a sample as required, and also another of the same size directly from the Pois(10) distribution, is

```
y1 = rpois(10000,4)
y2 = rpois(10000,6)
sample1 = y1 + y2
sample2 = rpois(10000,10)
```

   The two samples may be neatly compared as follows

```
plot(sort(sample2), sort(sample1), pch=3)
```

9. Appropriate **R** code to generate the sample is

```
z = rexp(10000,-log(2/3))
sample = 1 + floor(z)        # or use:  sample - ceiling(z)
```

10. Possible **R** code to generate a sample as the sum of n independent $\mathrm{U}(0, 1)$ random variables, and to verify approximate normality is, with $n = 6$,

```
y = rep(0,10000)
for(i in 1:6) y = y + runif(10000)
qqnorm(y)
```

   Note that even with this quite small value of n, the normal approximation is already very accurate.

11. Possible **R** code is

```
times = cumsum(rexp(250,2)) #larger sample than necessary
accept = (runif(250)<0.5*(1+sin(2*pi*times/100))) & times<=100
inhomtimes = times[accept]
plot(inhomtimes,1:length(inhomtimes),xlab='time',ylab='N(t)',pch=3)
```