Series of notes on Randomization – Part III

R techniques in generating random numbers

R has the ability to use a variety of random number generating algorithms or in its term RNG (random number generators). The default generator is Mersenne-Twister, developed by Makoto Matsumoto & Takuji Nishimura in 1997 with a cycle period of $2^{19937}-1$. The other RNG methods in the R program are Super-Duper, Wichmann-Hill, Marsaglia-Multicarry, Knuth-TAOCP-2002, Knuth-TAOCP and L'Ecuyer-CMRG, etc.

Hence if we use the function RNG(), we have the default Mersenne-Twister method. We can use RNGkind() command to show this:

```
> RNGkind()
[1] "Mersenne-Twister" "Inversion"
```

The above output shows two items listed. The first is the standard number generator and the second is the one Inversion used for normal distribution generation.

We can also use RNGkind() to alter the RNG algorithm. For example if we want to change it to Super-Duper method, we write:

```
> RNGkind(kind='Super',normal.kind='Box')
> RNGkind()
[1] "Super-Duper" "Box-Muller"
```

In order to return to its defaults, we write:

```
> RNGkind('default')
> RNGkind()
[1] "Mersenne-Twister" "Box-Muller"
> RNGkind('default','default')
> RNGkind()
[1] "Mersenne-Twister" "Inversion"
>
```

Notice that if we were not to enter two defaults in the RNGkind() command, we would retain the previously altered algorithm.

As we have mentioned earlier, random number generation follows the basic principle of uniform probability distribution for equal chances. R language for uniform distribution function is runif(). It refers to:
unif(n,min=0,max=1) where n is the number of random variables, min and max are the lower and upper limits of the uniform distribution, respectively.

For example:

```
> runif(3,1,5)    # to generate 3 random numbers with [1,5] range
[1] 2.324228 4.257119 3.213694
```

If we were to write:

```
> runif(3)    # to generate 3 random numbers with default range
[1] 0.8547782 0.6827073 0.6220230
>
```

we would get 3 random numbers within [0,1] range

But, runif() does not give us the same random numbers every time we run a particular command. In some simulation exercise where we demonstrate something or testing and want to get the same thing time and time again, we can use the set.seed() command to do this:

```
> set.seed(1)    # 1 seed or starting point for RNG
> runif(1)
[1] 0.2655087
> runif(1)
[1] 0.3721239
> runif(1)
[1] 0.5728534
> set.seed(1)
> runif(3)
[1] 0.2655087 0.3721239 0.5728534
>
```

In here, we set the seed to 1 and then use three separate commands to create three random numbers. If we then reset the seed to 1 and generate three more random numbers (using only a single command this time), we get the same values.

We can also use the set.seed() command to alter the kind of algorithm using the kind= and normal.kind= instructions in the same way as we did when using the RNGkind() command:

```
> set.seed(1,kind='Super')
> runif(3)
[1] 0.3714075 0.4789723 0.9636913
> RNGkind()
[1] "Super-Duper" "Inversion"
>
> set.seed(1,kind='default')
> runif(3)
[1] 0.3721239 0.5728534 0.9082078
>
> RNGkind()
[1] "Mersenne-Twister" "Inversion"
```

In the above example, we set the seed using a value of 1 and altered the algorithm to the Super-Duper version. After using this to make three random numbers, we look to see what we have before setting the seed to 1 again but also resetting the algorithm to its default, the Mersenne-Twister.

How can we apply the above concept to randomized sampling?

For example, we have 100 cartons of cocoa butter fat for shipment and we need to sample 10% or 10 cartons for inspection. As usual, we first label all the cartons in running numbers from 1 to 100. We now use R language to tell us how to randomly sample from this consignment as below:

```
> Consignment=c(1:100)
> sample(Consignment,size=10)
 [1] 21 89 93 65 61 6 20 17 64 35
>
```

So, we have extracted 10 carton numbers randomly from the 100 assigned numbers.

Actually the sample() command contains of a default value: replace=FALSE. If we were to ask the following command with replace=TRUE, we may get an item to be selected more than once:

```
> Consignment=c(1:100)
> sample(Consignment, size=10, replace=TRUE)
 [1]  77  50  72 100  39  78  94  22  66  13
> sample(Consignment, size=10, replace=TRUE)
 [1] 27 39 2 39 87 35 49 60 50 19
>
```