Simulation

This part of the notes is complementary to the previous chapter where we saw how to deal with simple programming. Now the programming concepts in `S-Plus` will be employed to derive some simple simulation results.

The Weak Law of Large Numbers

It is well known that if $X_1, \ldots, X_n$ are independent and identically distributed random variables with finite expectation μ, then the weak law of large numbers states that

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \to \mu,$$

in probability, as $n \to \infty$. In particular if $X_1, \ldots, X_n$ are binary random variables with $P(X_i = 1) = p$, then

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \to p,$$

in probability, as $n \to \infty$. We will illustrate this results empirically by employing the following simple functions:

```
uniforms <- runif(500)    #generate 500 random variables
                          # from uniform in (0,1).
tosses   <- as.numeric(I(uniforms > 0.5))   # if a uniform is greater than 0.5
                                            # set 1, otherwise 0. Equivalent to
                                            # tossing a coin with probability
                                            # of success 0.5.
relfreq  <- cumsum(tosses)/(1:500)     # Take the relative frequency
                                       # cumsum is cumulative sum
plot(relfreq)                          # plot relative frequencies
abline(0.5,0)                          # Convergence
title(main="Illustration of the Weak Law of Large Numbers")
```

The results is illustrated in Figure 1 which clearly shows convergence towards 1/2.

A more instructive example is given by the following

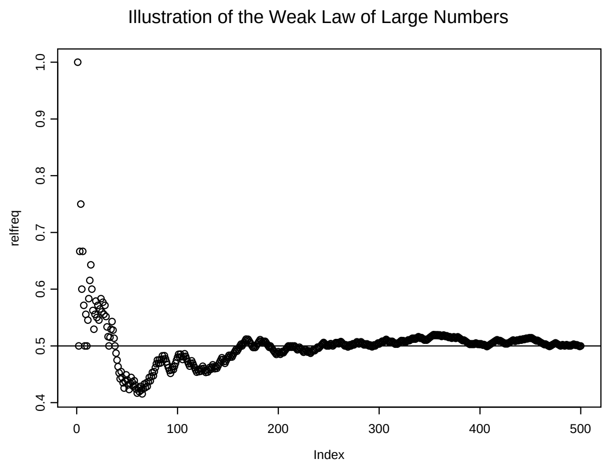

Figure 1: The Weal Law of Large Numbers for Binary Random Variables

```
par(mfrow=c(2,2))
for(rep in 1:4){
uniforms <- runif(500)
tosses   <- as.numeric(uniforms > 0.5)
relfreq  <- cumsum(tosses)/(1:500)
plot(relfreq)
abline(0.5,0)}
```

which will display a two by two plot of random sequences converging to 0.5.

Let us see what happens when the expectation of the random variables is infinite. A well known example of such a random variable is given by the Cauchy distribution

$$f(x) = \frac{1}{\pi(1+x^2)}, \quad x \in R,$$

whose expectation does not exist. The following functions together with Figure 2 demonstrate that the sequence of averages does not converge.

```
cauchys <- rcauchy(500)
xbar    <- cumsum(cauchys)/(1:500)

for(rep in  1:4){
cauchys <- rcauchy(500)
xbar    <- cumsum(cauchys)/(1:500)
plot(xbar)
abline(0,0)}
```

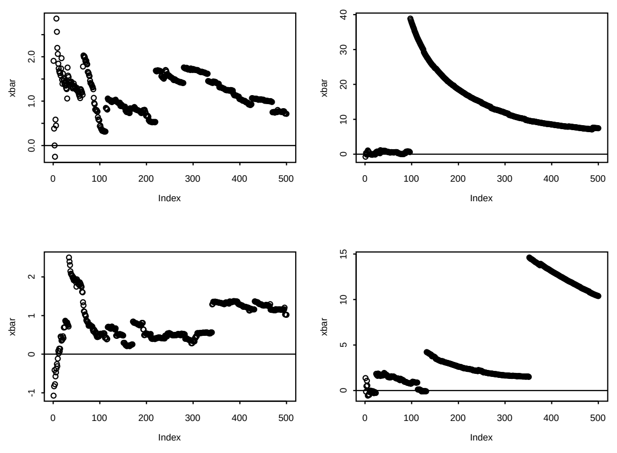

Figure 2: Sequence of Averages from Cauchy

The Central Limit Theorem

Suppose that $X_1, \ldots, X_n$ are random variables with mean μ and variance σ^2 which is assumed to be finite. Then the central limit theorem states that

$$\sqrt{n}(\bar{X} - \mu) \Rightarrow N(0, \sigma^2),$$

in distribution, as $n \to \infty$. Here the symbol N stands for the normal distribution. We will confirm empirically this result by simulation. Suppose that $X_1, \ldots, X_{100}$ are independent and identically distributed Poisson random variables with parameter $\lambda = 1$. Then $\mu = \sigma^2 = 1$ and therefore the central limit theorem states that

$$\mathsf{E}(\bar{X}) = 1$$

and

$$\mathsf{Var}(\bar{X}) = 1/100 = 0.01.$$

The following functions generates samples from the asymptotic distribution of the mean:

```
poisson.clt    <- function(k,n, parameter)  #function of k=number of sample,
                                            #n=sample size, parameter=lambda of
                                            #Poisson
{
  samples.mean   <- rep(NA, k)      #initialize the vector of means
  for (i in 1:k){                   #Do k times generation of sample size n
                                    # from Poisson with lambda=parameter
  samples.mean[i]   <- mean(rpois(n, lambda=parameter))
```

```
       }
  return(samples.mean)             #return the results
}
```

Here are the results when running this function:

```
> test.pois <- poisson.clt(200,100,1) #generate 200 means from 100 Poisson random
                                      #variables with lambda=1.
> summary(test.pois)                   #report minimum, Q1, median, mean, Q3
                                       #and maximum. The mean and median are 1.
 Min. 1st Qu. Median  Mean 3rd Qu. Max.
 0.69    0.93      1 1.001   1.072 1.31
> var(test.pois)                       #The variance is close to 0.01
[1] 0.009584601
> par(mfrow=c(1,2))                    #plot a histogram and a boxplot
> hist(test.pois)                      # of the results.
> boxplot(test.pois)
```

The numeric results together with Figure 3 illustrate empirically the central limit theorem.

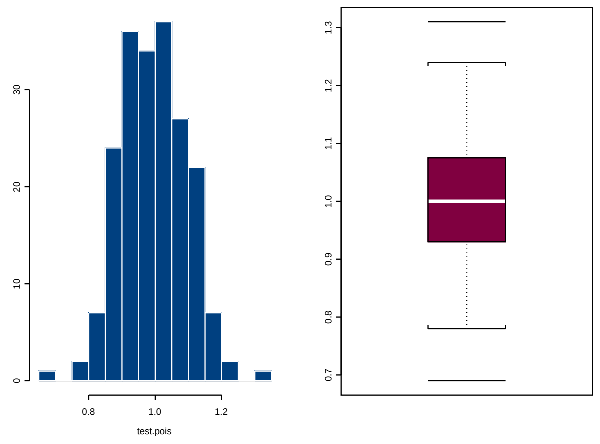

Figure 3: Central Limit Theorem for Poisson

It should be noted that the function `poisson.clt` is not the most efficient way of programming but it presents the general idea behind the calculations. For efficient use of loops in terms of memory and CPU time the function `lapply` is more suitable.

Monte Carlo Integration

Assume that X is a random variable with probability density function $f(x)$ and consider the problem of estimating an expectation of its function given that the latter exists. Specifically let δ be defined by the following

$$\delta = \int c(x)f(x)dx = \mathsf{E}_f\left[c(X)\right],$$

and assume that it exists and it is finite. There are several methods for evaluating δ–perhaps the most popular among statisticians are those which based on Monte Carlo integration. Accordingly, if $X_1, \ldots, X_n$ is a random sample from $f(x)$, then the weak law of large numbers implies that the following estimator

$$\hat{\delta} = \frac{1}{n}\sum_{i=1}^{n} c(X_i)$$

approaches δ with high probability as n tends to infinity.

Here is how S-plus can be used to evaluate such integrals. Suppose that X is a Beta random variable with probability density function

$$f(x) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

for $x \in (0,1)$ and suppose that we wish to evaluate the following two integrals:

$$\delta_1 = \int_{0.2}^{0.4} f(x)dx = \int I_[0.2, 0.4]f(x)dx,$$

where I is the indicator function, and

$$\delta_2 = \int \sin(x)e^{-x}f(x)dx$$

where we will assume that f stands for the Beta density with parameters 2.5 and 5. Then the following function is returning the desired results in matrix form.

```
estim.beta  <- function(k)   #k is the number of simulations
{
delta1      <- rep(NA, k)    #initialize delta1
delta2      <- rep(NA, k)    #initialize delta2
for (i in 1:k)               #Do k runs
{
x   <- rbeta(500, shape1=2.5, shape2=5) #generate 500 observations
                                        #from Beta with parameters 2.5 and 5.
delta1[i] <- mean(as.numeric(I(0.2 <x < 0.4))) #compute delta1 and
delta2[i] <- mean(sin(x)*exp(-x))            #delta2
}
return(cbind(delta1,delta2))              #return the results in matrix form
}
```

When we run the function we obtain

```
> delta.estim <- estim.beta(500)   #Do 500 simulations
> apply(delta.estim,2,mean)        #Get the means of each column
delta1    delta2
 0.23072 0.2172346
> apply(delta.estim,2,var)         #Get the variance of each column
 delta1        delta2
 0.0003627351 9.412981e-006
> sqrt(apply(delta.estim,2,var))  #Get the Monte Carlo standard error
 delta1      delta2
 0.01904561 0.003068058
```

Hence we see that δ_1 is estimated as 0.03086395 with a standard error of 0.0027 while δ_2 is estimated by 0.08337394 with a standard error of 0.0025.

Although there are several standard methods to generate pseudorandom outcomes for many distribution occasionally such a task can be quite demanding due to the functional form of the probability density function. Hence some other techniques might be used instead and one of the most popular simulation based methods is the so called *importance sampling* whose implementation is outlined below:

- Generate $Z_1, \ldots, Z_n$ i.i.d. data from a probability distribution function $g(z)$ whose support, say A, includes the support of $f(.)$.
- Upon noticing that

$$\begin{aligned}\delta &= \int c(z)\frac{f(z)}{g(z)}g(z)dz \\ &= \int c(z)w(z)g(z)dz = \mathsf{E}_g\left[c(Z)w(Z)\right],\end{aligned}$$

  with $w = f/g$, form the following estimator

$$\tilde{\delta} = \frac{1}{n}\sum_{i=1}^{n} c(Z_i)w(Z_i).$$

Apparently $\tilde{\delta}$ mimics the estimator $\hat{\delta}$ given by in the sense that expectation with respect to X following f is replaced by the corresponding expectation with respect to Z following g. It will be instructive to think how to use `S-Plus` to compute $\tilde{\delta}$ for the example given above when g is the uniform density.