Spring 2011 Geog 210C: Analytical Methods in Geography, Lab 2
Bootstrapping and Monte Carlo Simulation

Preliminaries: Consider the 2 data files ithaca.dat and canandaigua.dat available from the class Web page. Each of these files contains 4 columns: (i) day index from Jan 01-31, (ii) precipitation in inches, (iii) minimum temperature in degrees Fahrenheit, and (iv) maximum temperature. Load the 2 files into R workspace, and create a new larger data set by appending the 2nd file to the 1st, as: data = rbind(ithaca, canandaigua); notice that col 1 goes 1-62, the day column (2) has 1-31 two times. Now, create a more interesting maximum temperature data by adding 20 to the last 32 rows on col 4.
data[32:62,4]=data[32:62,4]+20; From now on, we will be referring to this max temperature data set as: tmax = data[,4].

Bootstrap from a sample data set:
1. A bootstrap sample from the data array tmax can be generated by: (i) generating N = 62 uniformly distributed integers in the interval [1,N] as: index1 = ceiling(N*runif(N,0,1)), and (ii) picking the sample data values that correspond to those integers: tmaxBoot1 = tmax[index1]. Repeat the above steps 2 more times to generate 2 other bootstrap samples, and compare the histograms (?hist) of the 3 bootstrap samples to that of the original data in array tmax.

2. You can generate M = 1000 sets of bootstrap samples, each of size N, as: IndexM = matrix(ceiling(N*runif(N*M)), ncol=M), TmaxBoot = matrix(tmax[as.numeric(IndexM)], ncol=M); Each column of array IndexM contains one bootstrap sample. Now compute the M means of the M bootstrap samples as: meanBoot = apply(TmaxBoot, 2, mean); (?apply). Display the histogram of these mean values as hist(meanBoot,breaks = 50, main=”Histogram of 1000 bootstrap-generated mean values”) and interpret it. Is the average of the bootstrap-derived mean values mean(meanBoot) close to the sample average mean(tmax) of the data in array tmax?

3. The procedure described above is automated in function boot, which can compute sample statistics other than the sample mean. The function boot is part of a separate package. See http://cran.r-project.org/ and search for boot, install the boot package. For example the same task above can be accomplished as: meanBootAuto = boot(tmax, statistic=boot.mean, R=M); Display the histogram of bootstrap sample means in meanBootAuto, and verify that it is similar to that generated in Requisite #2. What is the 3rd and 5th boot-strapped mean value provided by boot? These are located in meanBootAuto$t.

Tip: you may need this function

boot.mean <- function(x,i) {
  boot.sample <- x[i]  # bootstrapped sample
  m = mean(boot.sample)  # bootstrapped mean
  return(m)
}
4. Use function boot to generate M bootstrap values for the sample variance. Display the histogram of the bootstrap-generated sample variance values, and interpret it. Tip: use the function provided above, use var instead of mean.

5. Consider the bootstrap sample mean values stored in array meanBoot, and compute and interpret the following bootstrap-derived quantities: (i) the standard error of the sample mean: Note: make sure that the std.error function is loaded, if not find the package that contains it. (ii) the 95% confidence interval of the sample mean: quantile(meanBoot,0.975)-quantile(meanBoot,0.025), and (iii) the p-value for an observed sample mean of 44 degrees Fahrenheit? : hint for iii, how many values greater than 44?

6. What are the assumptions behind all the computational experiments conducted above?

Monte Carlo Simulation (Non-Parametric)

Preliminaries: In what follows, we will be using the augmented temperature data set stored in array tmax.

Monte Carlo simulation from a sample CDF:

1. Construct the sample CDF of the data in tmax as: ecdf_tmax = ecdf(tmax), and display the resulting CDF plot as stairs. Can you identify in that CDF plot the two modes seen in the corresponding histogram of the tmax-values?

2. Generate a column vector of N = 62 random numbers uniformly distributed in [0, 1] as: p = runif(N,0,1). Use function interp1 and the outputs from function ecdf generated above to compute the quantiles of the tmax-values associated with these simulated probabilities as: qSim = interp1(Fx, xS, p, method ="nearest"). You have just generated your first Monte Carlo sample from scratch!!!. Compute the mean and variance of the simulated values as mean(qSim) and var(qSim) and compare them to those of the original data in tmax. Check the agreement between the distribution of simulated values and that of the original data in tmax using via a QQ-plot as qqplot(tmax,qSim), and comment on the results. How can you generate another Monte Carlo sample of size N? Would you expect the statistics of simulated values to be the same from one Monte Carlo sample to another?

3. Repeat the above task with N = 1000, and N = 10000, and comment on your findings. Use different names for the array p and the array of resulting quantiles for each case.

Links between bootstrap and non-parametric Monte Carlo:

4. The default mode for function interp1 is linear interpolation. In our context, this means that the call to interp1 in Requisite #2 above, used linear interpolation to compute the quantiles associated with the simulated probability values in array p, based on those
probabilities in array $F_x$ associated with the sorted sample data in array $x_S$. Consider
now using nearest neighbor instead of linear interpolation in the call to interp1. Can you
anticipate what values would be generated using such a modified call to interp1? Would
you anticipate similarities between these simulated quantiles and the bootstrap sample
tmax$Boot_1$ generated in Requisite #1 in the previous page? Can you think of links
between non-parametric Monte Carlo simulation and bootstrap?

**Parametric Distributions and Random Numbers**

1. Explore the different probability distributions available in R, runif, rnorm, rexp,
rbinom;

2. Generate $N=100$ values from a Gaussian distribution with mean 10 and std deviation
5 as $y_{Sim1} = \text{rnorm}(N1,10,5)$; Plot the CDF of the simulated values as plot(ecdf(ySim1),
main="CDF of $N=100$ simulated values"), and compute their mean $m_Y = \text{mean}(y_{Sim})$
and std deviation $s_Y = \text{std}(y_{Sim})$. Repeat the same tasks with $N1=1000$ and $N2=10000$,
and note any differences in the mean and std deviation of the simulated values.

Do the statistics of the simulated values and their CDF converge to the expected
statistics and CDF as the sample size increases?

2. Generate $N=99$ values from a std normal distribution as $y_{Sim} = \text{rnorm}(N, 0,1)$;
and create a quantile-quantile plot between these simulated values and the
corresponding quantiles of the theoretical std Gaussian distribution. Comment on
the utility of such a plot. Hint: You need to evaluate the theoretical quantiles of a
std Gaussian distribution for a probability axis discretized into $N=99$ intervals, and
plot these theoretical quantiles against the sorted simulated values. This can be
done as:

\[
p = ((1:N)-0.5)/N
q\text{Theory} = \text{qnorm}(p,0,1)
\]
plot(sort(ySim),qTheory, pch=19, cex=0.6)

4. Show via simulation that: (i) $E\{a + bX\} = a + bE\{X\}$, for any constants $a$ and $b$,
and for any distribution, (ii) $E\{X^2\} \neq (E\{X\})^2$ for any distribution, and (iii)
$\text{Var}\{a + bX\} = b^2\text{Var}\{X\}$, for any constants $a$ and $b$, and for any distribution.