2011 BIOSTAT 615/815 Homework #5

Due is Tuesday March 29th, 08:30AM. No hard copy. Emails only.

Problem 1. Simulation of biased coin

Write a C++ program that creates a simulated input data of `biasedCoinHMM` problem in Homework 4. The hidden state $S = \{0, 1\}$ represents fair (F) and biased (B) states respectively, and the observation $O = \{0, 1\}$ represents head (H) and tail (T), respectively. The HMM parameters are

$$\begin{aligned} \pi &= (0.90.1)^T \\ \Pr(S_t = 0|S_{t-1} = 0) &= 0.95 \\ \Pr(S_t = 1|S_{t-1} = 1) &= 0.8 \\ \Pr(O_t = 0|S_t = 0) &= 0.5 \\ \Pr(O_t = 0|S_t = 1) &= 0.9 \end{aligned}$$

Your program need to generate input files similar to `simulCoinInput.txt` posted in the homework 4, but it should vary based on the seed you're providing. For example, an example command line arguments should look like

```
user@host~/ > biasedCoinSimulator 100   # simulates 100 observations with seed obtained from timestamp
user@host~/ > biasedCoinSimulator 100 1 # simulates 100 observations with starting seed 1
```

Write two separate programs (biasedCoinSimulator1.cpp, biasedCointSimulator2.cpp) that does the same thing, but one using `rand()` function build in the compiler, and the other using Mersenne Twister algorithm built in the `boost` library. Email (1) the pair of codes (2) and example a pair of outputs of 100 observations (from each program) to the instructor, with brief comments. Do not print the hard copy of the code.

The example output should look like (but not identical to)

```
1  F  H
2  F  H
3  F  H
4  F  T
5  F  T
6  F  T
7  F  H
8  F  T
9  F  H
10 F  H
```

Problem 2. Simulation of multivariate normal random variables

Write a C++ program that simulates a set of random variables following multivariate normal distribution with specified covariate matrix as input files. Let V is $n \times n$ covariance matrix, then each row of the output files contain n values of MVN random variables following $N(0, V)$. You need to use both `boost` package and `Eigen` package for accomplishing the

```
// A guide for simulating multivariate normal distribution
#include "Matrix615.h"  // take it from the course web site
#include <ctime>
#include <Eigen/SVD>
#include <Eigen/Core>
#include <Eigen/Cholesky>
#include <Eigen/Dense>
#include <boost/random/variate_generator.hpp>
#include <boost/random/mersenne_twister.hpp>
#include <boost/random/normal_distribution.hpp>
#include <boost/lexical_cast.hpp>
```

```
using namespace Eigen;

int main(int argc, char** argv) {
  if ( argc != 3 ) {
    std::cerr << "Usage: simulMVN [covariance_matrix] [# of samples]" << std::endl;
    abort();
  }

  // read matrix from file
  Matrix615<double> W;
  W.readFromFile(argv[1]);
  if ( W.numRows() != W.numCols() ) {
    std::cerr << "The input file is not exactly square" << std::endl;
    abort();
  }
  int n = W.numRows();
  int m = boost::lexical_cast<int>(argv[2]);

  MatrixXd V;
  W.copyTo(V);

  LLT<MatrixXd> llt(V); // compute Cholesky decomposition
                        // you may want to use llt.matrixL() or llt.matrixU() later on
  // *** FILL IN - declare random variable generator for normal distribution









  VectorXd x(n);
  MatrixXd Y(m,n);
  // *** FILL IN - use x to generate size-n i.i.d. random vectors
  //               and use the llt (Cholesky decomposition) to convert it to
  //               multivariate normal distributed vectors
  //               each row of Y contains the MVN-distributed random variables









  // FILL IN : calculate Ve, an empirical covariance matrix of Y across m sets of samples
  VectorXd mY = Y.colwise().mean();       // calculate columnwise mean
  Y.rowwise() -= mY;                      // substract mean for each row
  // Y.rowwise() -= Y.colwise().mean();   // This does not work
  // *** FILL IN BELOW TO CALCULATE THE EMPIRICAL COVARIANCE MATRIX
  MatrixXd Ve =
  std::cout << "Empirical covariance matrix is " << std::endl;
  std::cout << Ve << std::endl;
}
```

An example input covariance matrix is

```
1.0  0.5  0.5  0.5  0.5  0.0  0.0  0.0  0.0   0.0
0.5  1.0  0.5  0.5  0.5  0.0  0.0  0.0  0.0   0.0
0.5  0.5  1.0  0.5  0.5  0.0  0.0  0.0  0.0   0.0
0.5  0.5  0.5  1.0  0.5  0.0  0.0  0.0  0.0   0.0
0.5  0.5  0.5  0.5  1.0  0.0  0.0  0.0  0.0   0.0
0.0  0.0  0.0  0.0  0.0  1.0  0.5  0.5  0.5   0.5
0.0  0.0  0.0  0.0  0.0  0.5  1.0  0.5  0.5   0.5
0.0  0.0  0.0  0.0  0.0  0.5  0.5  1.0  0.5   0.5
0.0  0.0  0.0  0.0  0.0  0.5  0.5  0.5  1.0   0.5
0.0  0.0  0.0  0.0  0.0  0.5  0.5  0.5  0.5   1.0
```

Run the program with the covariance matrix with 10,000 sets of random vectors (m=10,000) to report the emprical covariance matrix. Email the code and the outputs (in pdf or text files) to the instructor

Problem 3. Integration of multivariate normal distribution

Write an R (or C++) program that integrates the following standard normal density function using Monte Carlo methods

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

between $-50 \leq x \leq 50$. In other words, we want to compute the following value θ

$$\theta = \int_{-50}^{50} f(x)dx$$

Implement the followng five Monte Carlo Methods to estimate θ

1. The Crude Monte Carlo Method
2. Rejection-based Monte Carlo Method
3. Importance Sampling using the Cauchy distribution
4. Importance Sampling using the Student t-distribution with df=10
5. Importance Sampling using the Student t-distribution with df=100

In each case, sample $n = 10,000$ random values (10,000 pairs of random values for Rejection-based method), sampled from uniform distribution or other specified distribution (in the case of importance sampling) to calculate the estimated theta $\hat{\theta}$. Repeat the experiment 100 times, and calculate the mean and SE of $\hat{\theta}$ for each of the five method.

E-mail the full souce code to the instructor, with a summary of mean and SE of $\hat{\theta}$ as a separate attachment (pdf or text file). Do not turn in any hard copy

Note that you are not allowed to use functions related to Gaussian CDF such as `pnorm()`. The list of functions you are allowed to is limited to

- `runif`
- `rcauchy`
- `rt` (df=10, 100 only)
- `dnorm`
- `dcauchy`
- `dt` (df=10, 100 only)