# >>> Assignment #2 for Simulation (CIS 4930) <<<

## >>> SOLUTIONS <<<

This assignment covers material from the second week of class lecture.

## Problem #1

Answer the following probability theory questions:

a) Roll a pair of dies (dice). What is the probability that the sum of the dots is 6? What is the probability of getting a 1-1, 2-2, or 3-3?

There are 36 outcomes of which 5 result in a 6 (that is, roll a 1-5, 2-4, 3-3, 4-2, or 5-1). So, the probability of rolling a 6 is 5/36. There are 3 outcomes that are the pair 1-1, 2-2, or 3-3. So, the probability is 3/36 = 1/12 to roll a1-1, 2-2, or 3-3.

b) A lottery is based on 50 balls each with a number on it (1, 2, …, 50) placed in a bucket and thoroughly mixed-up. Six balls are pulled from a bucket without replacement. What is the probability of getting 1-2-3-4-5-6? What is the probability of getting 47-23-7-13-2-36? Explain why the probabilities are the same or different (whichever the case may be). Note that in a lottery it does not matter what order the numbers appear in, just that the set of selected numbered are all "pulled".

The solution for any sequence of 6 unique values (in any order) is:
$$\left(\frac{6}{50}\right)\left(\frac{6-1}{50-1}\right)\left(\frac{6-2}{50-2}\right)\left(\frac{6-3}{50-3}\right)\left(\frac{6-4}{50-4}\right)\left(\frac{6-5}{50-5}\right)=\frac{1}{\binom{50}{6}}=6.293\times10^{-8}$$

The probability of any sequence of 6 unique numbers is the same due to independence – the probability of any given remaining number being pulled is independent of the previous number pulled, thus 1-2-3-4-5-6 is just as likely 47-23-7-13-2-26, or any other sequence of 6 unique numbers.

c) Assume that the probability of a PC being "dead on arrival" (DOA) due to an independent manufacturing defect is 1 in 100. If you order 300 PCs for (say) your business, what is the probability that the first PC you unpack and install is DOA?

If the probability of a PC being DOA is 0.01 (1 in 100) and this probability is independent of previous events, then the probability of any PC being DOA, be it the 1st in a batch of 300 or (say) the 57th in a batch of 4 million, is 0.01.

## Problem #2

Design of reliable data transfer methods is a challenging problem. The simplest such method is to simply transfer all data blocks $N$ times ($N > 1$) each. Assume that the probability of a data block being corrupted is $p$, errors occur independently, and corrupted data blocks can always be detected. Define a success rate, $S$, to be the probability that a given data transfer is successful (i.e., that at least one of the $N$ blocks transferred is not corrupt, or bad).

a) If $p$ is 0.05, what is the minimum $N$ for $S$ to be no less than 99.99%, 99.999%, and 99.9999%? Note that $N$ must be an integer! **Hint:** What is the probability that all $N$ blocks transferred are bad? Knowing this you can easily find the probability of one or more blocks transferred being good (not bad).

The probability of all $N$ blocks being bad (that is, corrupted) is $p^N$. So, the probability of one or more good blocks (successful transfers) in N transfers is $1-p^N$. We need to solve for $N$ for $0.9999 = 1-(0.05)^N$, $0.99999 = 1-(0.05)^N$, and $0.999999 = 1-(0.05)^N$. Doing so we get $N$ = 3.07,

3.84, and 4.61. Clearly, $N$ has to be integer so the final answer is $N = 4$, 4, and 5. We can also get to the solution by numerically solving for $1 - p^N$ for $p = 0.05$ and $N = 1, 2, 3, 4,$ and $5$:

$1 - 0.05^1 = 0.95$, $\quad 1 - 0.05^2 = 0.9975$, $\quad 1 - 0.05^3 = 0.999875$, $\quad 1 - 0.05^4 = 0.99999375$, $\quad$ and $1 - 0.05^5 = 0.999999969$

We see that to get four 9s we need $N = 4$, to get five 9s we need $N = 4$, and to get six 9s we need $N = 5$.

b) For a given $S$ and $p$, solve for $N$. That is, find a general expression for $N$ in terms of $S$ and $p$.

The general expression is $S = 1 - p^N$, we can solve for $N$ as

$$1 - S = p^N$$
$$\log(1 - S) = \log(p^N)$$
$$\log(1 - S) = N \cdot \log(p))$$
$$N = \text{ceil}\left(\frac{\log(1 - S)}{\log(p)}\right)$$

Where the ceil function effectively rounds-up the result to the next highest integer $N$.

c) A Monte Carlo simulation of this problem is `as3_2_1.c` (available on the class source code page for download here: `http://www.csee.usf.edu/~christen/class3/source3.html`). Run this simulation (you will need to change the parameter values in the program – you cannot just compile and run it) and see if the answers you get match with what you computed for (a). Show your run time results. How do the results from this simulation vary as a function of NUM_ITER. Why?

See Appendix A for execution output of as3_2_1.c for $N$ = 1, 2, 3, 4, and 5 (and PR_BAD = 0.05 and NUM_ITER = 1000000). The results are "close" to the expected (theory), but not exact. The reason for not being exact is that this is a statistical experiment – only if it were run for infinity trials would it give the exact (theory) result. The error is due to "experimental variability".

## Problem #3

A Monte Carlo simulation of the dice roll problem we discussed in class (that is, the probability of rolling a pair) is `diceSim2.c` (available here: `http://www.csee.usf.edu/~christen/class3/source3.html`). Run it can convince yourself that the probability of rolling a pair is 1/6. Modify this program to model three die (say, red, blue, and green) and the event of rolling a triple (so, 1-1-1, 2-2-2, …, 6-6-6) and determine the probability of this event occurring. You are now writing (albeit simple) simulations!

The solution (program diceSim3.c) is in Appendix B. The execution shows that the probability of rolling a triple is 2.77%, which is (very close to) 6/126 = 1/36 as expected by theory. Note the update to the header and otherwise correct usage of C style guidelines.

## Problem #4

Assume you flip a fair coin 4 times – call this a trial. You can have 0, 1, 2, 3, or 4 heads appear in each trial. What is the expected (mean) number of heads that will appear given an infinite number of trials? What is the variance of the number of heads that will appear? Plot the probability mass function for the number of heads that appear.

Let $X$ be the random variable for the number of heads. Easiest is to enumerate all possible events (but, we could also note that X = 0, 1, 2, …, 4. heads is binomially distributed).

```
Event        Number of heads
------------------------------
T T T T            0
T T T H            1
T T H T            1
T T H H            2
T H T T            1
T H T H            2
T H H T            2
T H H H            3
H T T T            1
H T T H            2
H T H T            2
H T H H            3
H H T T            2
H H T H            3
H H H T            3
H H H H            4
```

So, we have 1 instance of 0 heads, 4 instances of 1 head, 6 instances of 2 heads, 4 instances of 3 heads, and 1 instance of 4 heads. The expected number of heads E[X] is then:

$(1/16) \cdot 0 + (4/16) \cdot 1 + (6/16) \cdot 2 + (4/16) \cdot 3 + (1/16) \cdot 4 = 2$

The variance $\sigma^2$ is $E[X^2] - E[X]^2$ so we need to calculate $E[X^2]$:

$(1/16) \cdot 1^2 + (4/16) \cdot 1^2 + (6/16) \cdot 2^2 + (4/16) \cdot 3^2 + (1/16) \cdot 4^2 = 5$

And, thus $\sigma^2 = 5 - 2^2 = 1$.
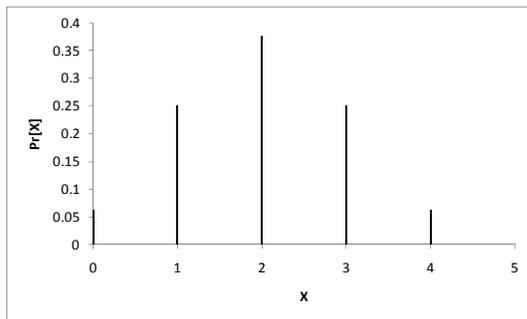


## Problem #5

Derive the expression for variance of the continuous uniform probability distribution (we derived the expression for mean in class). Show your work. Using `genunifc.c` from the Christensen tools page generate 1 million samples with $a = 1.0$ and $b = 10.0$. Find the mean and variance of the samples (that is, the experimental mean and variance). Compute the relative error of the experimental mean and variance compared to the theory mean and variance (that is, as calculated from the derived expressions)? **Hint:** the program `summary1.c` from the Christensen tools page can be used to compute summary statistics for a series of values (be sure to read the header in the program to learn how to use it!).

$$\sigma^2 = \left[ \int_a^b x^2 \cdot \left( \frac{1}{b-a} \right) dx - \left[ \int_a^b x \cdot \left( \frac{1}{b-a} \right) dx \right]^2 \right]$$

$$\int_a^b x^2 \cdot \left( \frac{1}{b-a} \right) dx \rightarrow \frac{a^2}{3} + \frac{a \cdot b}{3} + \frac{b^2}{3}$$

Subtract $E[X]^2 = ((a+b)/2)^2$ and you get…    $\sigma^2 = \frac{(a-b)^2}{12}$

Here are the results from `genunifc.c` and `summary1.c`:

```
■  Administrator: work                                                    ─ ▢ ✕

c:\work>bcc32 genunifc.c
Borland C++ 5.2 for Win32 Copyright (c) 1993, 1997 Borland International
genunifc.c:
Turbo Link  Version 2.0.68.0 Copyright (c) 1993,1997 Borland International

c:\work>genunifc
------------------------------------ genunifc.c -----
-   Program to generate continuous uniform RVs        -
------------------------------------------------------
Output file name ====================================> x
Random number seed ==================================> 1
Min value (continuous) ==============================> 1.0
Max value (continuous) ==============================> 10.0
Number of values to generate ========================> 1000000
------------------------------------------------------
-   Generating samples to file                        -
------------------------------------------------------
------------------------------------------------------
-   Done!                                             -
------------------------------------------------------

c:\work>summary1 < x
---------------------------------------------- summary1.c -----
   Total of 1000000 values
      Minimum  = 1.000004 (position = 551245)
      Maximum  = 10.000000 (position = 1310)
      Sum      = 5500270.538353
      Mean     = 5.500271
      Variance = 6.743041
      Std Dev  = 2.596737
      CoV      = 0.472111
------------------------------------------------------------------

c:\work>
```

The theory mean is 5.5 and the theory variance is 6.75 (from the above formulas). So, the relative errors for this set of generated values are:

$$100 \cdot \left( \frac{5.500271 - 5.5}{5.5} \right) = 4.927 \times 10^{-3} \quad \text{percent}$$

$$100 \cdot \left( \frac{6.743041 - 6.75}{6.75} \right) = -0.103 \quad \text{percent}$$

Which are very small errors in the context of performance modeling.

## Appendix A - Results for problem #2(c):

```
===============================================================
==       *** Results from data transfer simulation ***      ==
===============================================================
=   MODEL INPUT:
=     Number of transfer iterations     = 10000000
=     Number of blocks per transfer     = 1
=     Probability of a block being bad   = 0.050000
===============================================================
=   MODEL OUTPUT:
=     Success percentage = 95.005600 %
===============================================================


===============================================================
==       *** Results from data transfer simulation ***      ==
===============================================================
=   MODEL INPUT:
=     Number of transfer iterations     = 10000000
=     Number of blocks per transfer     = 2
=     Probability of a block being bad   = 0.050000
===============================================================
=   MODEL OUTPUT:
=     Success percentage = 99.749220 %
===============================================================


===============================================================
==       *** Results from data transfer simulation ***      ==
===============================================================
=   MODEL INPUT:
=     Number of transfer iterations     = 10000000
=     Number of blocks per transfer     = 3
=     Probability of a block being bad   = 0.050000
===============================================================
=   MODEL OUTPUT:
=     Success percentage = 99.987490 %
===============================================================


===============================================================
==       *** Results from data transfer simulation ***      ==
===============================================================
=   MODEL INPUT:
=     Number of transfer iterations     = 10000000
=     Number of blocks per transfer     = 4
=     Probability of a block being bad   = 0.050000
===============================================================
=   MODEL OUTPUT:
=     Success percentage = 99.999270 %
===============================================================


===============================================================
==       *** Results from data transfer simulation ***      ==
===============================================================
=   MODEL INPUT:
=     Number of transfer iterations     = 10000000
=     Number of blocks per transfer     = 5
=     Probability of a block being bad   = 0.050000
===============================================================
=   MODEL OUTPUT:
=     Success percentage = 99.999970 %
===============================================================
```

## Appendix B – Program for simulation of rolling a triple

```c
//======================================================== file = diceSim3.c =====
//=  Simulation of rolling three die and getting a triple                     =
//=============================================================================
//=  Notes:                                                                   =
//=    1) This program is for assignment #2 for CIS 4930 (Simulation) for     =
//=       Summer 2011. This is an example of a Monte Carlo simulation.        =
//=    2) See NUM_ROLL and TRACE in the #define section                       =
//=    3) Ignore two warnings on always false and unreachable code            =
//=---------------------------------------------------------------------------=
//=  Build: bcc32 diceSim3.c, cl diceSim3.c, gcc diceSim3.c                   =
//=---------------------------------------------------------------------------=
//=  Execute: diceSim3                                                        =
//=---------------------------------------------------------------------------=
//=  Sample execution:                                                        =
//=                                                                           =
//=    >>> The simulation is running...                                       =
//=    ==============================================================         =
//=    ==   *** Sim of rolling three die and getting a triple ***   ==        =
//=    ==============================================================         =
//=    =  MODEL INPUT:                                                        =
//=    =    Number of rolls  = 10000000                                       =
//=    ==============================================================         =
//=    =  MODEL OUTPUT:                                                       =
//=    =    Percent triples  = 2.773180 %                                     =
//=    ==============================================================         =
//=---------------------------------------------------------------------------=
//=  History: KJC (05/24/11) - Genesis (from diceSim2.c)                      =
//=============================================================================
//----- Include files ---------------------------------------------------------
#include <stdio.h>           // Needed for printf()
#include <stdlib.h>          // Needed for rand()

//----- Constants -------------------------------------------------------------
#define    FALSE      0      // Boolean false
#define     TRUE      1      // Boolean true

#define NUM_ROLL 10000000    // Number of iterations to simulate
#define    TRACE    FALSE    // Number of blocks transferred per iteration

//===== Main program ==========================================================
void main(void)
{
  int    redDie;          // Value of red die (1, 2, ..., 6)
  int    blueDie;         // Value of blue die (1, 2, ..., 6)
  int    greenDie;        // Value of green die (1, 2, ..., 6)
  int    tripleCount;     // Count of event redDie == blueDie == greenDie
  int    i;               // Loop counters

  // Output a "running" banner
  printf(">>> The simulation is running... \n");
```

```
  // Run the simulation for NUM_ROLL dice rolls
  tripleCount = 0;
  for (i=0; i<NUM_ROLL; i++)
  {
    // Roll the three die
    redDie = (rand() % 6) + 1;
    blueDie = (rand() % 6) + 1;
    greenDie = (rand() % 6) + 1;

    // Output values of dice is TRACE is enabled
    if (TRACE == TRUE)
      printf("redDie / blueDie / greenDie = %d / %d / %d \n",
        redDie, blueDie, greenDie);

    // Check if a pair and increment pairCount if so
    if ((redDie == blueDie) && (blueDie == greenDie))
      tripleCount++;
  }

  // Output results
  printf("================================================================ \n");
  printf("==   *** Sim of rolling three die and getting a triple ***   == \n");
  printf("================================================================ \n");
  printf("=  MODEL INPUT:                                                 \n");
  printf("=    Number of rolls  = %d    \n", NUM_ROLL);
  printf("================================================================ \n");
  printf("=  MODEL OUTPUT:                                                \n");
  printf("=    Percent triples  = %f %% \n", 100.0 * tripleCount / NUM_ROLL);
  printf("================================================================ \n");
}
```