Chiteri, Martin Akolo **Student ID:** 1231525 *CIV E 606 Assignment 3* ***23rd February, 2016***

Question ONE

a) Hand simulation to mimic the erection of a wall that requires 6 pallets of bricks

#	Event Description	EVENT LIST TNow	(Mins) Duration	Event end processing time		TNow	CHRONOLOGICAL LIST Processed Event	Fill Pallet Stack / Available#	Production
1	~~Labourer supply pallet~~	0	2	2		0	-	- , 3 nil	-
2	~~Labourer supply pallet~~	2	2	4		2	Labourer supply pallet	1 fil, 2 nil	
3	~~Mason 1 remove pallet~~	2	1	3		2	Labourer supply pallet	2 fill, 1 nil	
4	~~Mason 1 lay bricks~~	3	8	11		3	Mason 1 remove pallet	1 fill, 2 nil	
5	~~Labourer supply pallet~~	4	2	6		4	Labourer supply pallet	2 fill, 1 nil	
6	~~Mason 2 remove pallet~~	4	1	5		4	Labourer supply pallet	1 fill, 2 nil	
7	~~Mason 2 lay bricks~~	5	8	13		5	Mason 2 remove pallet	1 fill, 2 nil	
8	~~Labourer supply pallet~~	6	2	8		6	Labourer supply pallet	2 fill, 1 nil	
9	Labourer supply pallet	8	2	10		8	Labourer supply pallet	3 fill, 0 nil	
10	~~Mason 1 remove pallet~~	11	1	12		11	Mason 1 lay bricks	2 fill, 1 nil	1 / 11
11	~~Labourer supply pallet~~	12	2	14		12	Mason 1 remove pallet	3 fill, 0 nil	
12	~~Mason 1 lay bricks~~	12	8	20		12	Mason 1 remove pallet	3 fill, 0 nil	
13	~~Mason 2 remove pallet~~	13	1	14		13	Mason 2 lay bricks	2 fill, 1 nil	2 / 13
14	Labourer supply pallet	14	2	16		14	Labourer supply pallet	3 fill, 0 nil	
15	~~Mason 2 lay bricks~~	14	8	22		14	Mason 2 remove pallet	3 fill, 0 nil	
16	~~Mason 1 remove pallet~~	20	1	21		20	Mason 1 lay bricks	2 fill, 1 nill	3 / 20
17	~~Labourer supply pallet~~	21	2	23		21	Mason 1 remove pallet	3 fill, 0 nil	
18	~~Mason 1 lay bricks~~	21	8	29		21	Mason 1 remove pallet	3 fill, 0 nil	
19	~~Mason 2 remove pallet~~	22	1	23		22	Mason 2 lay bricks	2 fill, 1 nil	4 / 22
20	Labourer supply pallet	23	2	25		23	Mason 2 remove pallet	3 fill, 0 nil	
21	~~Mason 2 lay bricks~~	23	8	31		23	Mason 2 remove pallet	3 fill, 0 nil	
22	~~Mason 1 remove pallet~~	29	1	30		29	Mason 1 lay bricks	2 fill, 1 nil	5 / 29
23	Labourer supply pallet	30	2	32		30	Mason 1 remove pallet	3 fill, 0 nil	
24	Mason 1 lay bricks	30	8	38		30	Mason 1 remove pallet	3 fill, 0 nil	
25	Mason 2 remove pallet	31	1	32		31	Mason 2 lay bricks	2 fill, 1 nil	6 / 31

b) Statistics for the state variables from hand simulation

i) *The available space on the scaffold to place pallets*:

Add up all the available space throughout the simulation

= (3 + 2 + 1 + 2 + 1 + 2 + 2 + 1 + 1 + 1 + 1 + 1 + 1 + 1) = 20

Divided by the number of events scheduled = 20 / 25 = 0.8 or **80%**

ii) *The time the laborer spends waiting to serve the masons*

These are the times a laborer has to wait for the space on the pallet to become available so as to supply more. The time slots are:

Between 10 – 12 mins = 2 mins

Between 12 – 14 mins = 2 mins

Between 16 – 21 mins = 5 mins

Between 25 – 30 mins = 5 mins

Total = 14 minutes

Divide by 4 (number of instances laborer delays occurred) = **3.5 minutes**

iii) *The production rate of the operation*

6 / 31 = **0.193549**

This is the based on the time taken to lay the last pallet by the second mason when building a wall with six pallets

Question TWO

a) Generate 100 samples from a two parameter Weibull distribution i.e Weibull(1, 5) using the Inverse Transform Method

Sample source code attached:

```
public static partial class Formulas
{
	public static System.Boolean Formula(Simphony.General.Execute Element)
	{
		// Parameters for the Uniform distribution
		const int UNIFORM_LOW = 0;
		const int UNIFORM_HIGH = 1;

		// Parameters for the Weibull distribution
		const double GAMMA = 1.0; // Scale
```

```
            const double kSHAPE = 5.0; // Shape

            double x = 0.0; // Inverse of the Weibull CDF distribution
            double y = 0.0;


            // Begin processing for a hundred random deviates
            for (int counter = 0; counter < 100; counter++)
            {

                  /* 1) Sample y from the Uniform Distribution as a random number
                  with values between 0 and 1 */
                  y = SampleUniform(UNIFORM_LOW, UNIFORM_HIGH);

                  /* 2) Set x to be the inverse of the Cummulative Distribution Function
                  of the Weibull distribution which is  1 - e * - ( x / gamma )^k */
                  x = 1 - System.Math.Log(System.Math.Pow((y / GAMMA), kSHAPE));

                  // 3) Deliver x
                  TraceLine(System.String.Format ("({0:0000}) => {1}", counter+1, x));
            }

            return default(System.Boolean);
      }
}
```

b) Generate 100 samples from a Beta distribution i.e Beta(4, 3, 32, 73) using the Acceptance Rejection Method

Shape parameters are α = 4.0 and β = 5.0

Evaluate the Beta function (denoted by Γ) as a factorial:

α = 4 and β = 3

$$B(\alpha, \beta) = \frac{\Gamma(\alpha) - \Gamma(\beta)}{\Gamma(\alpha + \beta)} = \frac{(\alpha-1)!\,(\beta-1)!}{(\alpha+\beta-1)!} = \frac{(4-1)!-(3-1)!}{(4+3-1)!} = \frac{1}{60}$$

The value of c (highest possible value of y on the interval 32 to 73) is determined to be **2.07**, from an analytical tool

Sample source code attached:

```
public static partial class Formulas
{
      public static System.Boolean Formula(Simphony.General.Execute Element)
      {
            // Parameters to the Uniform distribution
            const int UNIFORM_LOW = 0;
```

```
            // const int UNIFORM_HIGH = 1;

            // Two parameters for the beta distribution, Low and High
            // const double ALPHA = 4.0;
            // const double BETA = 3.0;
            const double LOW = 32.0;
            const double HIGH = 73.0;

            double x = 0.0; // A uniform random variate defined on the PDF f(x) with interval [a, b]
            double X = 0.0; // A random variable with PDF function f(x) defined on the interval [a, b]
            double y = 0.0; // A uniform random variate on the interval [0, c]

            double c = 0.0; // Highest value of y from f(x)

            // Maximum value of f(x) defined for x in the interval[a, b] (from 32 - 72)
            c = 2.07; // 2.4576;

            // Begin processing for a hundred random deviates
            for (int counter = 0; counter < 100; counter++)
            {
                  x = SampleUniform(LOW, HIGH);
                  y = SampleUniform(UNIFORM_LOW, c);

                  X = 30 * x * System.Math.Pow((1 - x), 4);

                  // Find out if the point y falls on or below the PDF curve
                  while ( y > X )
                  {
                        // 1) Generate random numbers that are uniformly distributed
                        x = SampleUniform(LOW, HIGH);
                        y = SampleUniform(UNIFORM_LOW, c); // A value of y drawn from the sample

                        X = 30 * x * System.Math.Pow((1 - x), 4); // Represents a likely value of y
                  }

                  // If so then deliver x
                  TraceLine(System.String.Format ("({0:0000}) =>  {1}", counter+1, x));

            } // End for

            return default(System.Boolean);
      }
}
```

Question THREE

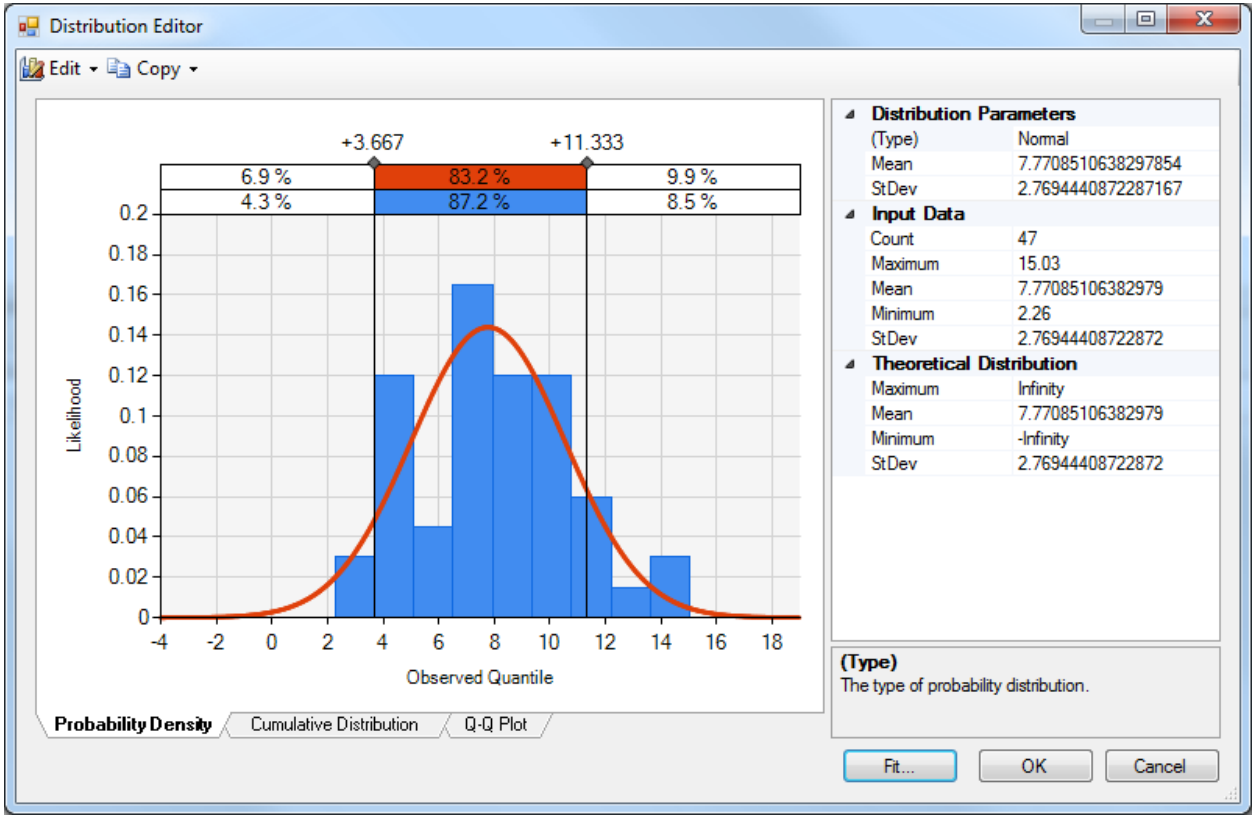
Using the method of moments to fit the statistical data in csv format

i) Normal Distribution

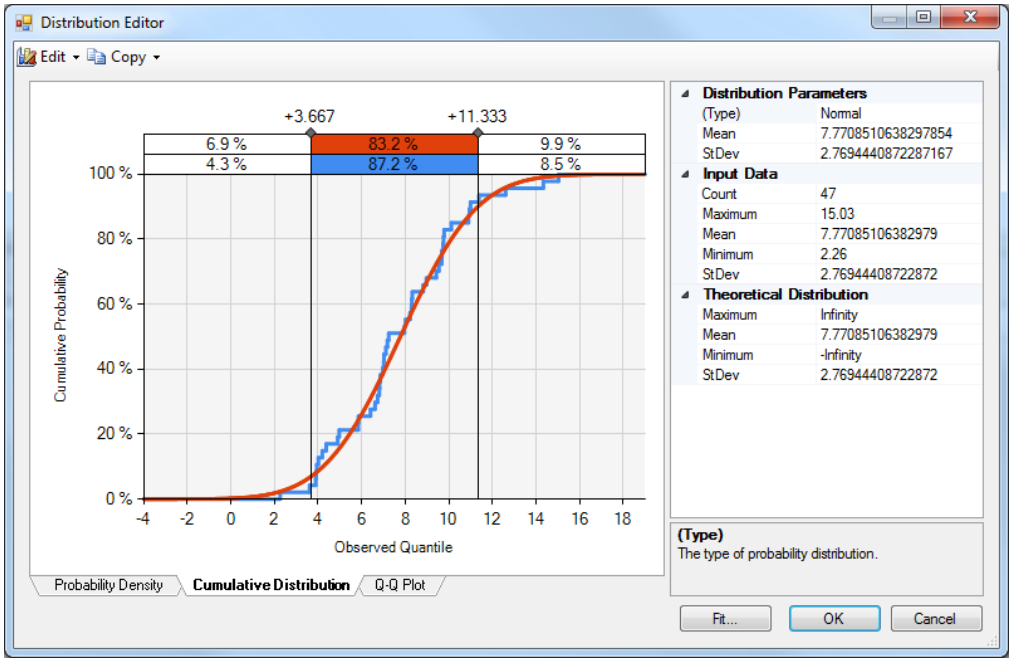
Mean = 7.7709

Standard deviation = 2.7694

PDF curve

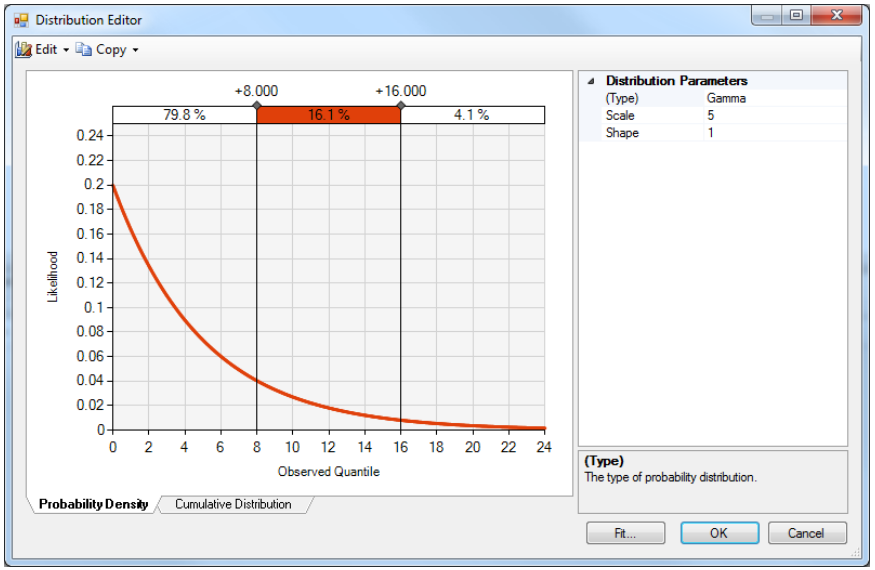


CDF curve

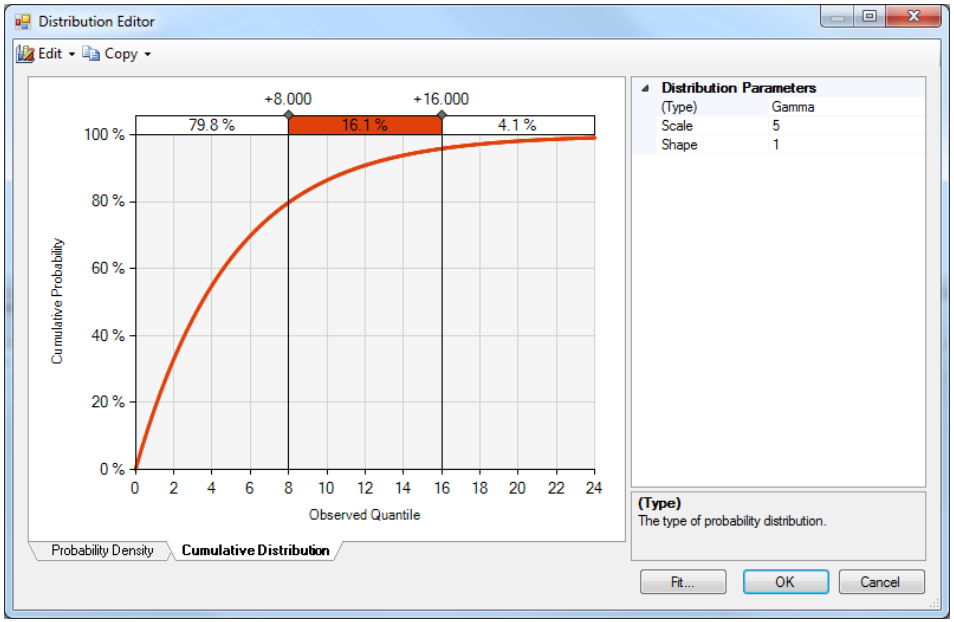


ii) Gamma Distribution

PDF Curve

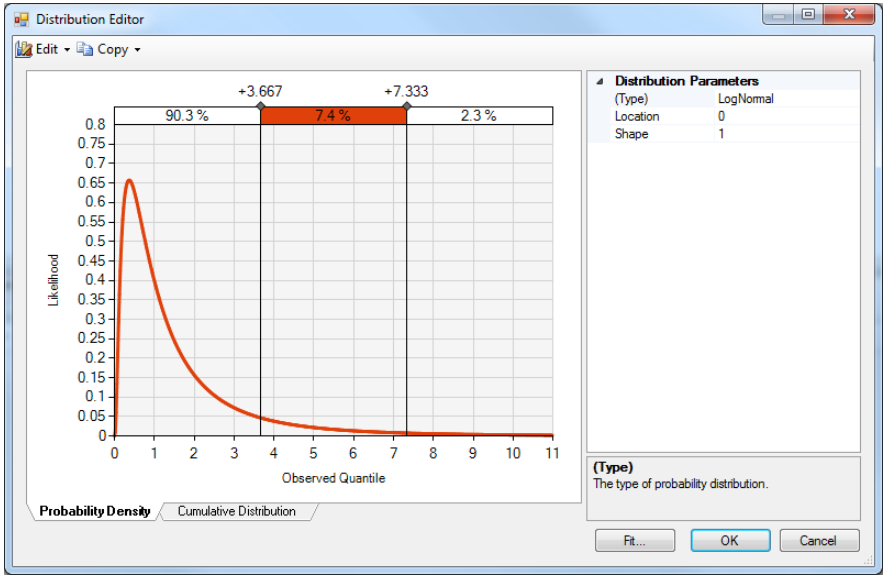


CDF curve

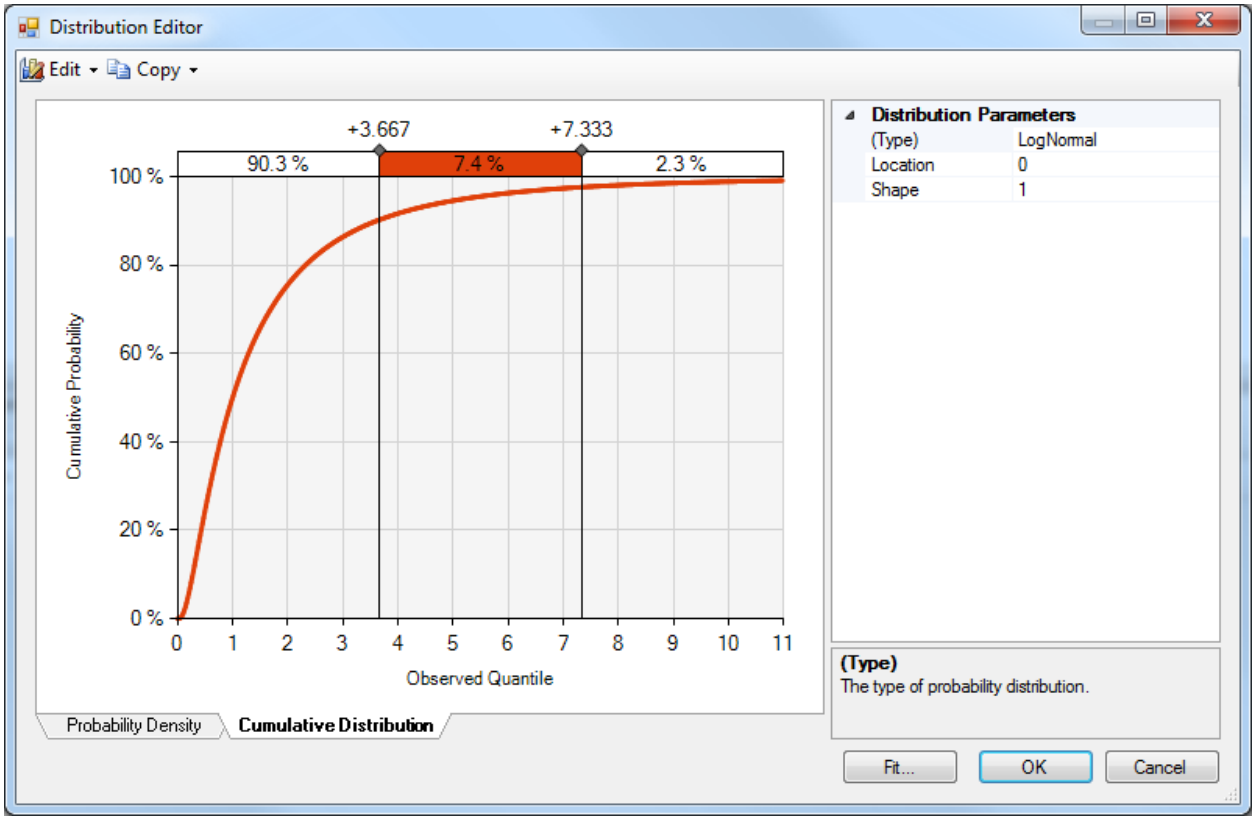


iii) Log normal Distribution

PDF Curve



CDF curve


Distribution Editor
Edit
Copy
+3.667
+7.333
90.3 %
7.4 %
2.3 %
100 %
80 %
60 %
40 %
20 %
0 %
Cumulative Probability
0
1
2
3
4
5
6
7
8
9
10
11
Observed Quantile
Probability Density
Cumulative Distribution
Distribution Parameters
(Type) LogNormal
Location 0
Shape 1
(Type)
The type of probability distribution.
Fit...
OK
Cancel