Distribution View: a tool to write and simulate distributions

Abstract

In our work we present a tool to write and simulate distributions. This tool allows to write mathematical expressions which can contain not only functions and variables, but also statistical distributions, including mixtures. Each time the expression is evaluated, for all inner distributions, is generated a value according to the distribution and is used for expression value determination. The inversion method can be used in this language, allowing to generate all distributions that have an expression for cdf inverse. The variables in the language allow the generation of several correlated distributions.

To illustrate the advantages of using distribution view we present two applications: One in Project Risk Management, compares the PERT method with Simulation alternative; The other in Statistics, compares the Power of Randomization Test with the power of Student-t Test , using the set of Marron-Wand distributions.

J. Silva Coelho[1], Fernando Branco[2], T. Azinheira Oliveira[3]

[1]CESUR/IST; Universidade Aberta, jcoelho@univ-ab.pt

[2] ULHT and CECME, Universidade Aberta, branco.fernando@netcabo.pt

[3] CEAUL and Universidade Aberta, toliveir@univ-ab.pt

Keywords: distribution generation, simulation, PERT, Power Tests, Randomization tests

1. Introduction

Uncertainty is part of almost, if not all, problems in our reality. Distributions allow us to quantify our knowledge over an uncertain variable, and mainly are used to describe observed random variables, and also in simulations to generate values for the random variables. In the first case we try to match a standard distribution to a set of observed values, and in the second case we select a standard distribution to generate a set of values.

Unfortunately, most of the times a standard distribution does not fit the data, and a non standard distribution is currently needed. For those cases we could define a particular density probability function, but that could be quite hard to do. We propose an alternative method to define a distribution based in standard distributions. The definition of a distribution is a mathematical expression which can contain not only functions and variables, but also statistical distributions, including mixtures.

We provide a generator and an application DistributionView[1] that accepts the definitions and generates values according to the distribution. Basically, each time the expression is evaluated, for all inner distributions, a value is generated according to the distribution and it is used for expression value determination. The application resemble a calculator that

[1] http://jcoelho.m6.net/freeware/DistributionView.html

accepts distributions, ready to use in a specific case by an engineer, and it is possible to call the generator inside a programming language, to be used by researchers.

The language is limited to distributions that can be generated from standard distributions, but since the inversion method can be used in this language, this is not a large limitation since all distributions that have an expression for cdf inverse can be generated. Also, the presence of variables in the language allow the generation of several distributions, which can be correlated.

To verify the usability of the proposed language in a research environment, we present two applications in different areas: One in Project Risk Management, compares the PERT method with Simulation alternative; The other in Statistics, compares the Power of Randomization Test with the power of Student-t Test. The results, instances, and engines produced in the research applications are available in the materials webpage[2], among with the instructions to easily reproduce the experiments, with the same or different parameters.

The rest of the paper is organized in the following way: in section 2 we give the definition of the language and present some examples, in section 3 we present the first application that compares the PERT method with Simulation alternative, in section 4 we present the second application that compares the Power of Randomization Test with the power of Student-t Test , using the set of Marron-Wand distributions. Finally, we end in section 5 with some conclusions. In appendix I we show example code of using the DistributionView generator inside VBA, MatLab and R. In appendix II we present the Marron and Wand distributions density functions and definitions.

2. Language Definition

The language is composed by mathematical expressions, which can contain functions, variables and distributions. An expression can be written in two syntaxes, list or function. Next is an abbreviate BNF definition:

```
<distribution>::=
 <number>|<variable>|<list-symbol>|<function-symbol>|<binary-symbol>
<list-symbol>::='['<symbol><separator><args>']'
<function-symbol>::=<symbol>(<args>)
<binary-symbol>::=<distribution><binary-operator><distribution>
<args>::=''|<distribution><separator><args>
<separator>::=' '|';'
```

A symbol is either a mathematical function or a regular distribution. The number of arguments is dependent on the symbol, and each argument can be either a constant, a variable or another distribution. The two syntaxes can be used in the same distribution, and are also allowed the usual binary operators. The symbols allowed are either continuous distributions, discrete distributions, mathematical functions or constants. They are listed in table 1, with the binary operators.

[2] http://jcoelho.m6.net/edicao3.asp?pa=4998

Table 1 – Symbols and binary operators allowed

Symbols	Category
Uniform(min; max) Exponential(mean) Normal(mean; dev) Lognormal(mean; mode; min) Triangular(min; mode; max) Chi_square(degree) t_student(degree) F_distribution(d1; d2) Cauchy(location; scale) Fisher_Tippett(a; b) - extreme value Laplace(a) - double exponential Logistic(location; scale) Pareto(a; b) Weibull(a; b)	Continuous distributions
Binomial(N; p) Geometric(p) Negative_Binomial(r; p) Hipergeometric(Mp; Mq; N) Poisson(mean) RND(x1;x2;...;xn)	Discrete distributions
max; min; sum – 1 or more arguments mul; div; pow; exp; log; log10 – 2 or more sqrt; ceil; abs – 1 argument	Mathematical functions
sin; sinh; asin; cos; cosh; acos; tan; tanh; atan	Trigonometric functions
lt; gt; elt; egt; eq; or; and; not	Logic functions
+ * / ^ <= >= < > = || &&	Binary operators
PI; EULER	Constants
if, cond, set	Conditionals and variables

For example, the normal distribution with mean 0 and standard deviation 1 can be specified either “Normal(0;1)” or “[Normal 0 1]”. A negative exponential distribution with mean 10 can be written either “Exponential(10)” or “[Exponential 10]”. The Distributions can be the result of an expression that contains one or more regular distributions, for example “Exponential(10)+Normal(0;1)*Normal(0;1)/Uniform(1;2)” is a definition composed by 4 independent distributions. In figure 1 is the result of generating 10.000 values for these distributions.

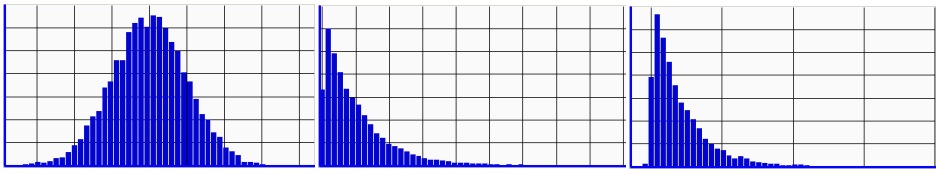


Figure 1 – Histogram of distributions: “Normal(0;1)”; “Exponential(10)”; “Exponential(10)+Normal(0;1)*Normal(0;1)/Uniform(1;2)”

Another example is the simulation of a dice throw, that has six discrete possibilities. We can use the distribution RND to generate values of equal probability: “rnd(1;2;3;4;5;6)”. The distribution of the sum of two dice throw is simple the sum of two distributions of a dice throw: “rnd(1;2;3;4;5;6)+rnd(1;2;3;4;5;6)”. Note that in this case, it is not required to calculate the probability of each possible result making easier to define the distribution. In figure 2 is shown the result of generating 10.000 values with these distributions.

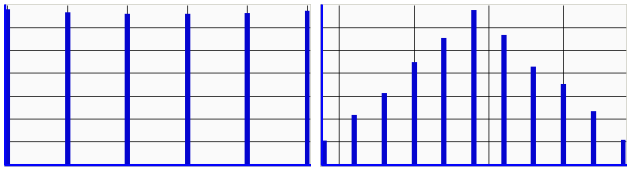


Figure 2 – Histogram of distributions: “rnd(1;2;3;4;5;6)”; “rnd(1;2;3;4;5;6)+rnd(1;2;3;4;5;6)”

The symbol **set** allows assigning distributions to variables. Variables are essential to generating more than one distribution at same time. The value returned by **set** is the last distribution. For instance, the distribution “[set Normal Normal(0;1) Triangular Triangular(-1;0;1) Uniform Uniform(-1;1)]” define three variables, called Normal; Triangular; Uniform, with distributions associated. The distribution “set(exp Exponential(10) max max(Exponential(10) Exponential(10)))” define a negative exponential distribution of average 10 and a distribution that is the maximal value of two negative exponential distributions of average 10. A third example is the distribution “set(pow Normal(0 1)^2 mul Normal(0 1)*Normal(0 1))” that we can see the difference between the square of a normal distribution of mean 0 and standard deviation 1, and the product of two normal distributions of mean 0 and standard deviation 1. In figure 3 are the resulting histograms.

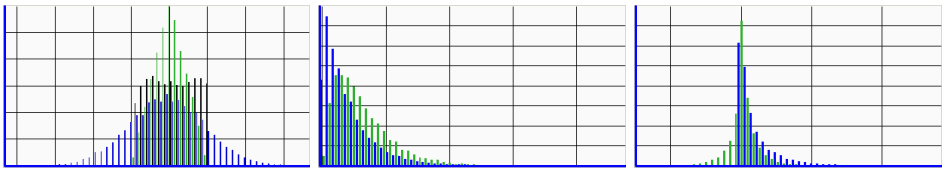


Figure 3 – Histogram of distributions: “[set Normal Normal(0;1) Triangular Triangular(-1;0;1) Uniform Uniform(-1;1)]”; “set(exp Exponential(10) max max(Exponential(10) Exponential(10)))”; “set(pow Normal(0 1)^2 mul Normal(0 1)*Normal(0 1))”

The symbol **set** can be used to generate correlated distributions, by using the values of a distribution in other distributions. For instance, the distribution “[set V1 Normal(10;1) V2 Normal(4;0.1)+Normal(2;0.1)*V1]” define two distributions (V1 and V2), and the second is close to 4+2*V1. We can give more examples: “set(DA Normal(10 1) max(Normal(10 1)+DA Normal(10 1)+max(DA Normal(10 1))))” or “set(V1 Normal(0 1)^3 V2 V1+Exponential(10))”. In figure 4 are the histograms of distributions in scatter plots to allow visualize the relation between the variables.

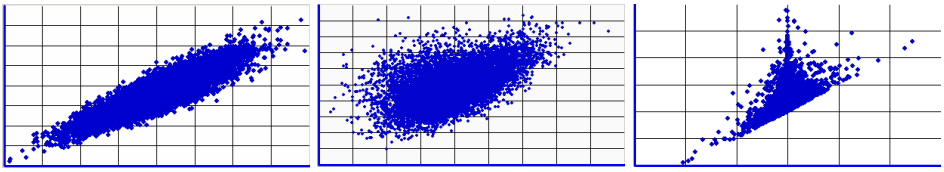


Figure 4 – Histogram of distributions "[set V1 Normal(10;1) V2 Normal(4;0.1)+Normal(2;0.1)*V1]"; "set(DA Normal(10 1) max(Normal(10 1)+DA Normal(10 1)+max(DA Normal(10 1))))"; "set(V1 Normal(0 1)^3 V2 V1+Exponential(10))"

The symbol **if(a;b;c)** return **b** if **a** is positive, otherwise return **c**. It allows some flow control like the symbol **cond(u;p1;d1;p2;d2;…;pn;dn)**, that add flexibility. In **cond** symbol the first distribution **u** is executed and also the distributions **p1** to **pn**, each time subtracting its value to the returned value of **u**. When this value became negative at a distribution **pk**, then **dk** is evaluated and returned its value. If **u=Uniform(0;1)**, then **pk** is the probability of **di** being returned. For instance, suppose that are required a discrete distribution with conditional probability: 0 (10%); 1 (15%); 2 (5%); 3 (30%); 4 (20%); 5 (20%). In this case the following expression generates the desired distribution "cond(Uniform(0;100) 10 0 15 1 5 2 30 3 20 4 20 5)". It is also possible to define a multi-modal distribution using this symbol. If we want to return a Normal(10;1) with 40% and Normal(15;2) with 60%, then the following distribution specifies that: "cond(Uniform(0;100) 40 Normal(10;1) 60 Normal(15;2))".

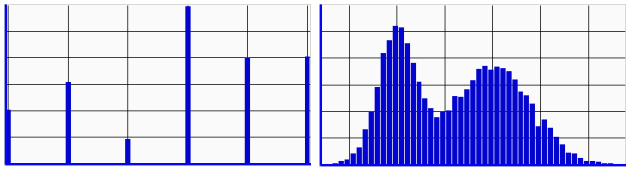


Figure 5 – Histogram of distributions: "cond(Uniform(0 100) 10 0 15 1 5 2 30 3 20 4 20 5)"; "cond(Uniform(0;100) 40 Normal(10;1) 60 Normal(15;2))"

3. PERT vs Simulation

In this section we present the first research application in Project Risk Management, that consists in analyzing the difference between the PERT method [Malcolm et al. 59] against the real distribution of the all network, obtained by simulation.

The PERT method consists in estimating the total duration project distribution by reducing the network to the activities in the critical path, turning the network in a serial one. In such a situation, the total duration is equal to the sum of all activity durations, and applying the theorem of central limit, the distribution of the total project duration is a normal distribution with average equal to the sum of the average of activities, and the variance is equal to the sum of the variances. Then, the total project duration distribution is used to calculate the probability of the project to be finished on time. This method is very popular since it is simple and it is included in Microsoft Project software[3]. We are interested in quantifying the error of using the PERT method instead simulating.

The total project duration distribution can be built in DistributionView language using a variable for the end time of each activity. The end time will be calculated by adding to the start time of the activity its processing time. The start time of an activity, if the scheduling strategy is to start as soon as possible (ASAP), is the maximal end time of the preceding

[3] http://www.microsoft.com/

activities, or is equal to zero if the activity has no preceding activities. The method PERT assume that all activities are in ASAP mode. Sometimes this strategy is not possible, mainly if we need to coordinate the project with outside suppliers, and obviously we need to specify the dates to allow the resources of each activity to be delivered in time. The earliest start time strategy (EST), sets the starting time of activities to the earliest possible assuming that all activities will be done with average time. To simulate this strategy we need to add the start time of each activity to the set of end times of the preceding activities:

```
[SET <activity-name> [SUM <processing-time>
  [MAX <activity-start-time> <end-time-of-preceding-activities>] …]
```

Since we want to return the highest value, because the project only ends when the last activity finish, we must return the maximal value of all activities.

For instance, suppose that you have the network in figure 6.

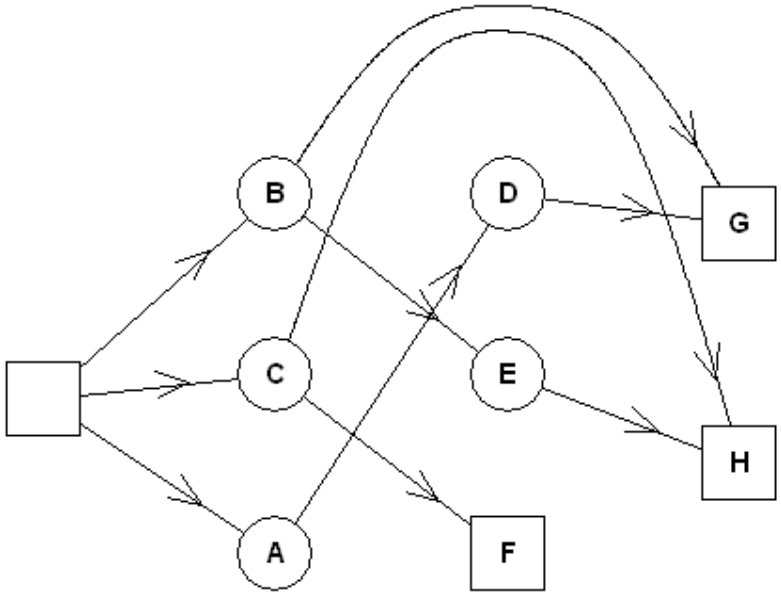


Figure 6 – Network with Activity-on-Node notation

Considering all activities with the same distribution Normal(10;1), the total duration of the project using ASAP strategy would be:

```
[SET A [SUM Normal(10;1) [MAX 0]]
[SET B [SUM Normal(10;1) [MAX 0]]
[SET C [SUM Normal(10;1) [MAX 0]]
[SET D [SUM Normal(10;1) [MAX 0 A]]
[SET E [SUM Normal(10;1) [MAX 0 B]]
[SET F [SUM Normal(10;1) [MAX 0 C]]
[SET G [SUM Normal(10;1) [MAX 0 B D]]
[SET H [SUM Normal(10;1) [MAX 0 C E]]
[MAX A B C D E F G H] ]]]]]]]]
```

In this case, with the PERT method the total project duration would be the distribution Normal(30;1.73), and simulating with the DistributionView for 10.000 runs the mean is 30.98 and the standard deviation 1.43. If we are interested to know the date that the project will end with 95% of certainty, we need to calculate the percentile of 95% for both distributions, and in this way we quantify the error of using PERT method. In the first case

the percentile of 95% is 32.8 and in the second case 33.0. The histogram of the project duration distribution can be seen in figure 7.

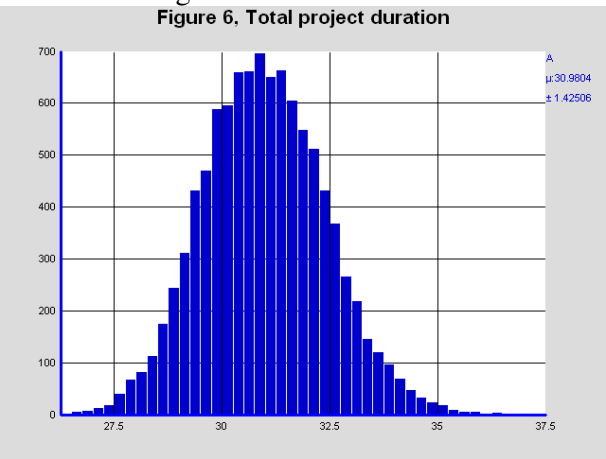


Figure 7– Histogram of total project duration of network in figure 6

This result is good for this project instance, but it is the general case? For answer this question we need to extend this test to a set of projects, which hope represents the all set of project instances. Of course, we cannot assert anything for the general case, but it will give us a better understanding.

We will use a benchmark of 480 networks with 30 activities each of the PSPLIB [Kolisch & Spreacher 97], to verify the difference between the distribution obtained by the PERT method and the simulated distribution.

It was done an engine to convert the files in PSPLIB format to the Distribution View format, using the duration of activities as the average, and setting the distributions of all activities as normal distribution with the same coefficient variation factor of 10% and 20%. The tests was done like the example above, but using 100.000 runs. The results are shown in the table 2.

Table 2 – Average results of the simulations done with 480 PSPLIB files of 30 activities

Coefficient variation factor	Method	Mean	Standard Deviation	Quantil 95%
10%	PERT	52,28	1,99	55,55
	ASAP	52,48	1,90	55,63
	EST	53,47	1,29	55,88
20%	PERT	52,28	3,98	58,82
	ASAP	53,06	3,64	59,16
	EST	54,83	2,58	59,61

We can notice that, for these networks, the mean increases in the simulation methods, but standard deviation decreases, even with the coefficient variation factor at 20%. The quantil of 95% is higher in ASAP and EST methods, that means if someone uses the PERT method to calculate the quantil of 95%, for project instances with characteristics close to this ones, will have an error in time units of 0,09 and 0,33 for ASAP and EST methods if the coefficient variation factor is 10%, and 0,34 and 0,80 for ASAP and EST methods if the coefficient variation factor is 20%. We cannot conclude anything solid with an empirical test, but we can at least make some conjectures: the ASAP has a higher mean, lower standard deviation and higher quantil than PERT and the same occur between EST and ASAP.

One could find strange the standard deviation to be lower in ASAP and even lower in EST than in PERT. To try to understand the relation we make a scatter plot between the standard deviation of PERT vs the standard deviation of ASAP, in figure 8. The scatter plot shows that in all tested cases the standard deviation is lower in ASAP than in PERT, since there is no cases below the identity function. This result can appear more normal if we think that PERT method consider only activities in critical path, and in such case the project can be late or earlier. In ASAP is considered all activities and therefore all paths, so is more likely the project duration to be late, since there are more activities, but there is less possibility the project duration to be earlier, since at least one path in the project will probably be late.

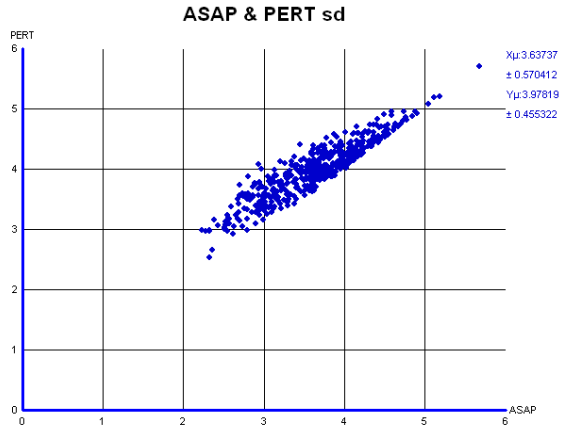


Figure 8– Scatter plot from ASAP vs PERT method in the standard deviation of the total project duration distribution

This application shows how easy is to use Distribution View to simulate a single project network by a project management, or used to simulate a set of project networks by a researcher. The test can be easily reproduced with the same parameters or different networks, different coefficient factor, or different quantil value, as explained in the materials webpage, already referred.

4. Randomization Test vs Student-t Test

In this section we present a second research application of Distribution View on randomization tests power evaluation. In our example two independent groups are to be compared considering the particular balanced case with 10 elements in each group: for each case we present the results for randomization test power versus t-test power.

It is well known that the t-test is the classical test used for the comparison of two independent groups, when the usual requests of random, normal and homocedastic samples are satisfied. In this case t-test is the most uniformly powerful test. However, in many situations, empirical research does not allow theoretical credibility for the requests of normality and homocedasticity. Also these requests are not easy to test particularly for cases of small size samples. The main problem is that in most of the situations, the researcher has not available a well specified random sample from population, but just a set of individuals or elements which he randomly assign to the conditions.

Randomization tests play an important rule on the research of particular situations, considering random distribution of individuals through the conditions, not random sampling in classical context. In traditional definition, a randomization test is a permutation test based

on random assignment to test the null hypothesis of no treatment effects, in a randomized experiment, and consists on a very important method for determining statistical significance.

The power of a test is known as the test capability to reject null hypothesis when null hypothesis is false, which means in this context, the capability of detecting a difference between groups, when it exists.

Fisher (1935) presents the first ideas for randomization tests, and Pitman (1937a, 1937b, 1938) develops some theoretical results. After that, many authors pay special attention to research on randomization tests, for example Kempthorne (1952), but at that time with difficult computation means, the interest was particularly on theoretical results development. With technological advance it is now easy to use computer simulation on tests generation, and some more recent important research on randomization tests applications can be found for example in Edgington (1995) and Manly (1997).

Although randomization tests are not appropriate to realize statistical inference from samples to populations, these tests can produce good results on the detection of treatment effects and on the establishment of local casualty relationships. However, results generalization on other sets can just be done under non-statistical basis.

In experimental data analysis using randomization tests, the test significance is calculated using a repetitive permutation data procedure, so the resultant significance is exact. It seams important to evaluate the power of the tests from data obeying different distributions. With the aim to represent some diversification on distributions found in literature, our simulations and analysis are based on the 15 Marron and Wand (1992) distributions, obtained with normal mixtures and frequently used as examples on robustness researches and analysis. These distributions density functions and definitions are presented in Appendix II.

For each distribution 100.000 samples were generated with 10 + 10 elements for each group, varying the effect size (ES) from -4 to 4, with 0.10 intervals (81 values). The effect size, defined by Cohen (1988), is the difference between the two population means, using population standard-deviation units:

$$ES = \frac{\mu_1 - \mu_2}{\sigma}$$

For each distribution and each ES value, we realized in each of the 100.000 samples both, the t-test and the randomization test. For the particular case of randomization test, once the combinations number is very big (184.756) we used a random sample with 1.999 of those combinations. On these choices, number of combinations and number of samples, we follow Westfall and Young (1993) suggestions. For each test the power was obtained, by calculating the samples proportion in which the result significance was equal or smaller than alpha=0.05.

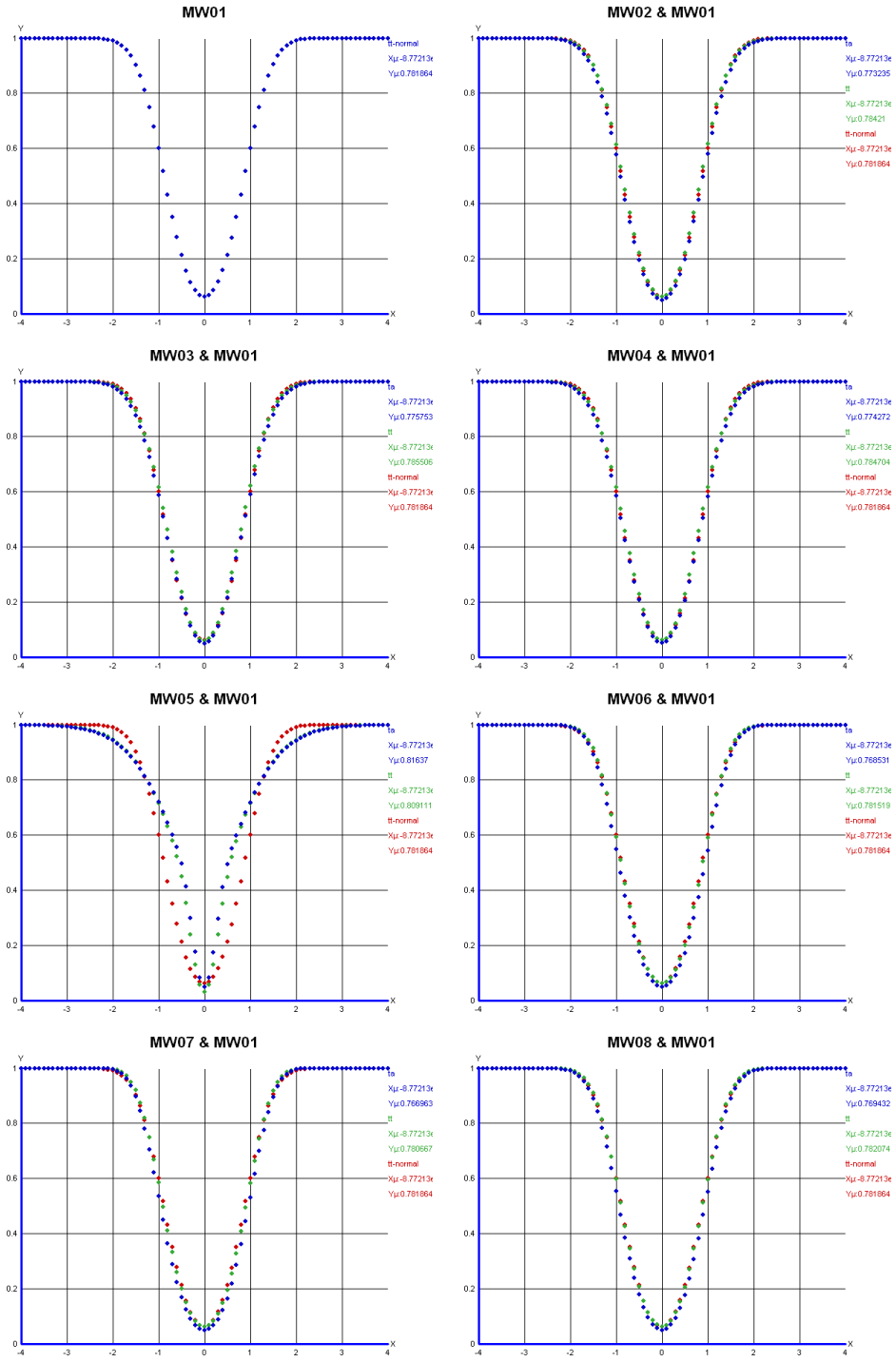
The 15 Marron-Wand distributions were transformed to Distribution View format, in a straightforward way. The table 3 clarifies this process with an example. Note that in the definitions was used the variance and in Distribution View format was used the standard deviation.

Table 3

Definition of #2 Marron-Wand	#2 in Distribution View format
Probability; Normal(mean; variance) 0.2; Normal(-0.3; 1.4400000) 0.2; Normal(0.3; 0.6400000) 0.6; Normal(1.0; 0.4444444)	**[Cond Uniform(0;10000)** **2000 Normal(-0,3;1,2)** **2000 Normal(0,3;0,8)** **6000 Normal(1;0,666667)** **]**

Since there are many things to do with each distribution, it was done an engine (PowerTest) in C++ that calls Distribution View inside C++ code, and this way can reproduce the results for any distribution in Distribution View format. The engine accepts an

instance file with the distribution, the number of elements of group 1 and group 2, the both values may differ, the value alpha, the number of random sample for the randomization test, and the effect size of the test. The engine returns the significance of the test to the left, right and bi-caudal tests, for both, randomization test and t-test. The bi-caudal test was used in the experiment. The test of the power of each distribution can be seen in figure 9.



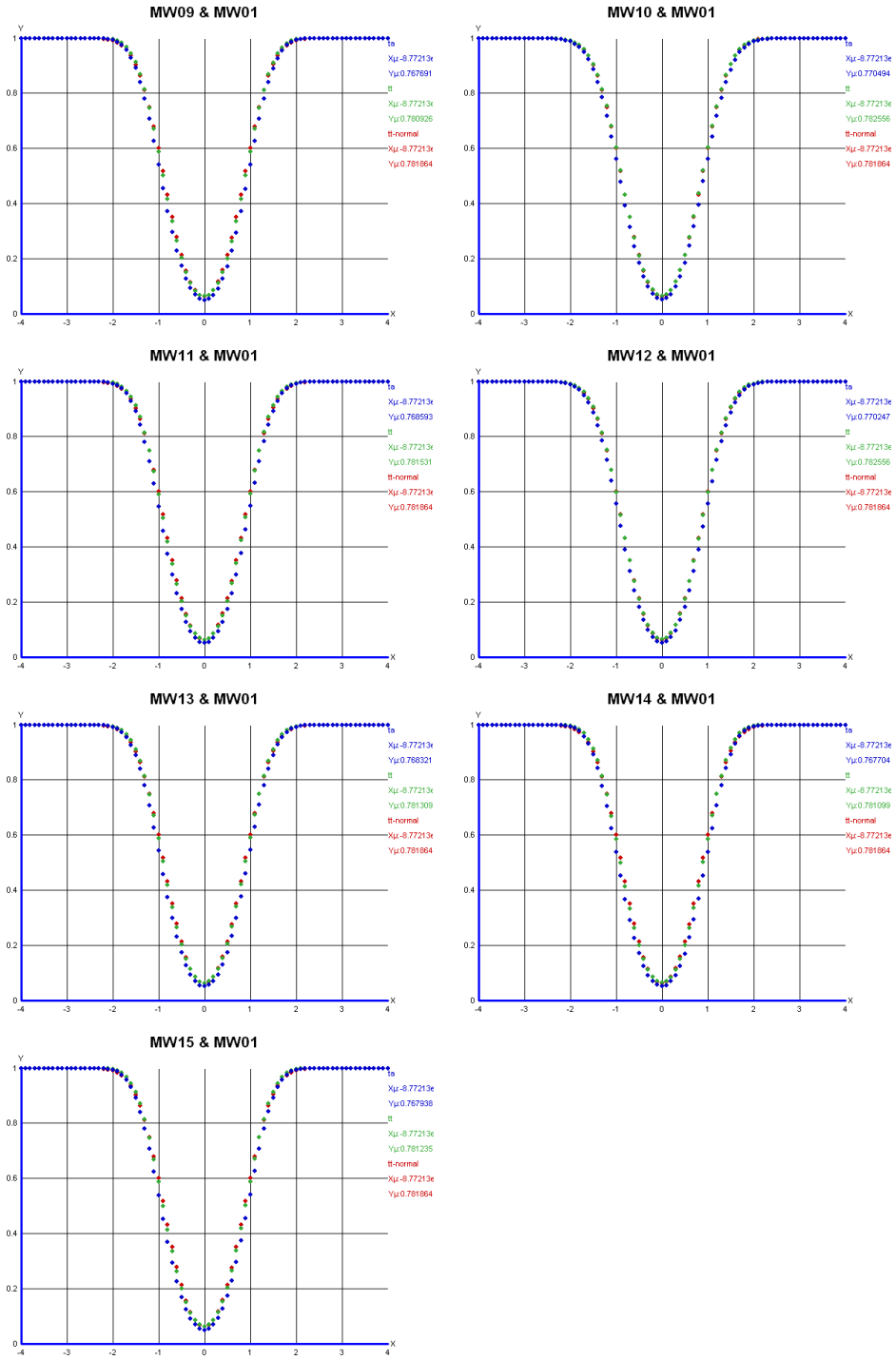


Figure 9

In a general way, we can say that in all the 15 analyzed distributions, the power curves of the randomization test and the t-test, are similar. The experience can be easily extended to other distributions and situations, in order to study, for instance, the impact of skewness, kurtosis, and other factors, on the power of the tests.

We make the experience available to be reproducible by anyone, in the materials webpage already referred. The reader is free to reproduce the experience with other

distributions in Distribution View format, using other values of effect size, alpha, and number of elements in each group, or even another size of the random sample used.

5. Conclusions

We described a method to define complex distributions in a simple way, and implemented an application ready to be used by an engineer or a teacher, like a calculator, and a generator that can be used inside a programming language, by a researcher.

We give two research applications examples, in different areas. In the first application on Project Risk Management, we present empirical tests that quantifies the average error of using the PERT method instead of using simulation, for a given set of project networks, and we make a conjecture about the difference between the total project duration distribution of PERT method and ASAP/EST methods. The application shows how easy is to use Distribution View to simulate a single project network by a project management, or used to simulate a set of project networks by a researcher.

In the second application in Statistics, we show how to evaluate the power of randomization test vs. Student t-test, considering that two independent groups are to be compared. The analysis is on the particular balanced case with 10 elements in each group. We show how to build the power of test for any distribution in Distribution View format, and we present results on the Marron-Wand distributions.

Both applications can be reproduced by other researchers using the materials webpage referred in section 1. The engines are open, so the tests can be reproduced using other parameters or other distributions.

References

- Abramowitz; Stegun, "Handbook of Mathematical Functions", Dover, New York, 1964
- J. Cohen, “Statistical power analysis for the behavioural sciences”, 2nd Ed. Hillsdale, NJ: Erlbaum, 1988
- E.S. Edgington, “Randomization tests”, 3rd Ed., New York: Marcel Dekker, Inc,1995
- R.A. Fisher, “ The Design of Experiments”, Edinburgh: Oliver & Boyd, 1935
- R. Guimarães; J. Cabral, “Estatística”, McGraw Hill, 1997
- TC Headrick, “Fast fifth-order polynomial transforms for generating univariate and multivariate nonnormal distributions”, Computational Statistics & Data Analysing 40 pp. 685-711, 2002
- O. Kempthorne, “ Design and Analysis of Experiments”, New York:John Wiley &Sons, 1952
- O. Kempthorne, “ The randomization theory of experimental inference”, Journal of the American Statistical Association, 50, 946-967, 1955
- R. Kolisch, A. Sprecher, “PSPLIB - A project scheduling problem library”, European Journal of Operational Research, Volume 96, Issue 1, Pages 205-216, 10 January 1997
- DG Malcolm; JH Roseboom; CE Clark; W Fazar, "Application of a technique for research and development program evaluation", Operations Research 7, pp. 646-669, 1959
- B.F.J. Manly, Randomization, Bootstrapping and Monte Carlo Methods in Biology, 2nd edn. London: Chapman & Hall, 1997.
- S. Marron & M. Wand, “Exact Mean Integrated Squared Error”, Annals of Statistics, 20, 712–736”, 1992.
- E.J.G. Pitman “Significance tests that may be applied to samples from any population.” Journal of the Royal Statistical Society, Suppt., 4, 119-130, 1937a
- E.J.G. Pitman “Significance tests that may be applied to samples from any population, II. The correlation coefficient test.” Journal of the Royal Statistical Society, Suppt., 4, 225-232, 1937b
- E.J.G. Pitman, “Significance tests that may be applied to samples from any population, III. The analysis of variance test.” Biometrika, 29, 322-335, 1938
- A Restrepo, AC Bovik, “On the generation of random numbers from heavy-tailed distributions.”, Proceedings of the IEEE 76 (7) pp. 838-840, 1988
- J Von Neumann, "Various Techniques Used in Connection with Random Digits." NBS Appl. Math. 12 pp. 36-38, 1951.

- Westfall, P. H. & Young, S. S., “Resampling-based multiple testing.” New York: John Wiley & Sons, 1993.

Appendix I

In this appendix is shown how to use Distribution View inside a programming language, for VBA, MATLAB and R. It is required the installation of the Distribution View.

VBA/Excel

In VBA, for example inside the Excel[4], we need to create a COM object “LOR.Distribution”. The following code exemplifies how this can be done.

```
Dim dist As Object
Set dist = CreateObject("LOR.Distribution")
dist.seed = 1 ‘ set seed number
dist.Definition = "Normal(0;1)"
For Each x In Selection
    dist.NextValue
    x.Value = dist.Value
Next
```

After the creation of the object “LOR.Distribution”, it can be used its members, like setting the seed number and the definition of the distribution. The cycle just places values of the distribution in the cells that are selected.

This example is not very useful if you want just to use the Excel as a worksheet and do not want to write any VBA code. For instance, you may want just to have a function (Rnd) that returns values according to the distribution that receive as an argument, like exemplified in Figure 10.

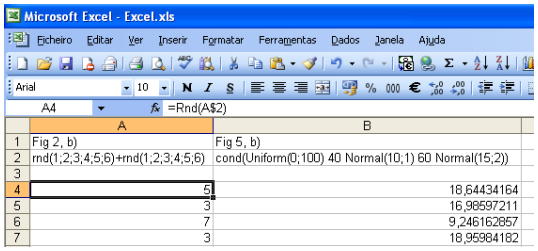


Figure 10

It is not good to create a COM object each time the function is used. The next code rearrange the previous code solving the problem. In a module we can insert the following code (a module is created for instance when you record a macro):

```
Dim dist As Object

Sub Init()
    Set dist = CreateObject("LOR.Distribution")
    dist.seed = 1 ' set seed number
    dist.Definition = "Normal(0;1)"
End Sub

Function Rnd(arg)
    If arg <> Empty Then
        If IsNumeric(arg) Then
            dist.seed = arg
        Else
```

[4] http://www.microsoft.com/

```
            dist.Definition = arg
        End If
    End If

    dist.NextValue
    Rnd = dist.Value
End Function
```

This way the object is created once in the Init subroutine, that must be called for instance in the Workbook_Open subroutine:

```
Private Sub Workbook_Open()
    Module1.Init
End Sub
```

The name of the module must be changed to the correct name. The function Rnd can be called in the worksheet as a function, and if it has an argument that is a number it will reset the seed number, otherwise will set the current distribution, returning a value of the current distribution in any case.

Language R

For language R[5] it is required the installation of the library RDCOMClient[6], for the following code work. The code can be copy/paste to the console of R:

```
library(RDCOMClient)
dist=COMCreate("LOR.Distribution")
dist[["Definition"]]="Normal(0;1)"
dist[["Seed"]]=0

Rnd<-function(arg="") {
  if(arg!="") {
    dist[["Definition"]]=arg
  }
  dist$NextValue()
  dist$Value()
}
```

Now, the function Rnd can be used to generate values according to a distribution defined in Distribtion View format. A session can be like this:

```
> Rnd("Max(Exponential(10);Exponential(10))")
[1] 10.23353
> Rnd("Max(Exponential(10);Exponential(10))")
[1] 9.733783
> for(i in 1:100) { a[i]=Rnd() }
> a
 [1] 37.055577 6.532512 20.895189 19.354479 12.560752 9.690168 14.861924
 [8] 4.960894 4.413664 6.852624 35.456110 29.579732 10.443467 41.501550
 [15] 48.629130 30.317320 3.854739 16.606489 16.632905 8.968086 4.831330
 [22] 2.492316 8.813771 5.554418 5.915057 22.362831 6.592735 16.984163
 [29] 32.484730 11.586687 20.698923 33.975156 16.835392 29.421565
22.111325
 [36] 17.867968 4.527751 17.980597 23.074650 1.723046 6.909455 12.429255
 [43] 4.831717 23.677794 27.884646 7.828116 7.501833 30.747653 15.295656
 [50] 17.733821 24.945318 19.988052 9.112212 6.906666 19.220544 13.761900
 [57] 30.439369 21.163865 5.290463 5.435791 17.463989 5.744752 47.040748
 [64] 4.750881 15.164997 27.250709 17.497735 10.119893 34.008231
11.447086
 [71] 15.382123 18.966725 4.645572 2.995799 8.531382 23.335539 11.802543
```

[5] http://www.r-project.org/

[6] http://www.omegahat.org/RDCOMClient/

```
 [78] 54.749213 17.036514 4.656288 23.883477 6.045892 36.744068 17.099009
 [85] 5.858834 10.152915 11.553047 28.211014 6.096965 7.156796 11.528382
 [92] 1.744044 18.956133 1.997445 7.453460 16.812183 9.392675 15.496670
 [99] 13.677202 29.658653
>
```

MATLAB

In MATLAB[7] we can define a function that returns a generator (a function) for any distribution given as an argument:

```
function dist = distribution(definition)
% This function returns a function that use a COM object
(LOR.Distribution)
% to generate distributions acoording to definition
% (see http://jcoelho.m6.net/freeware/DistributionView.html)

hCom=actxserver('LOR.Distribution');

hCom.Definition=definition;

dist=@f;

    function v=f(x)
        if nargin==1, hCom.Seed=x; end
        hCom.NextValue();
        v=hCom.Value;
    end
end
```

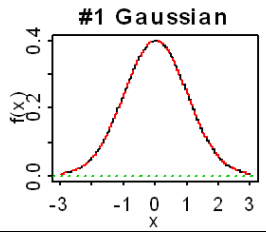
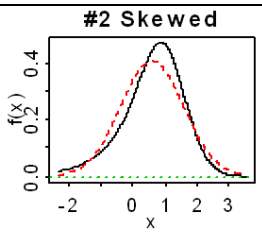
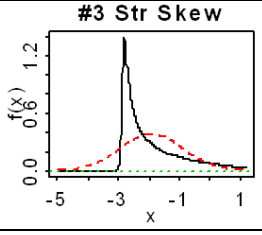
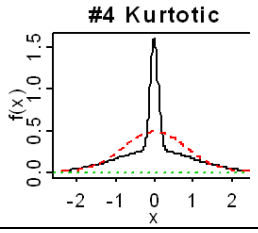
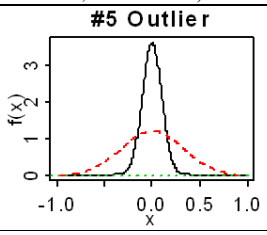
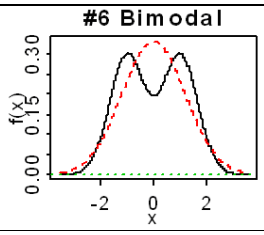
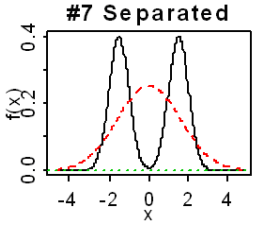
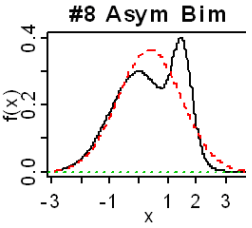
This function can be very convenient in MATLAB, for instance:

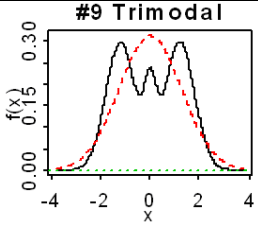
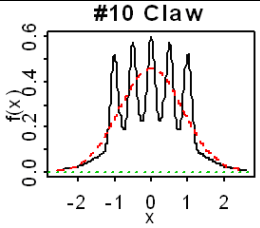
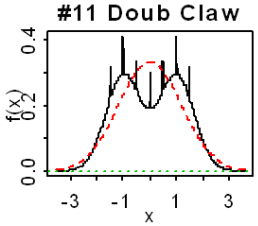
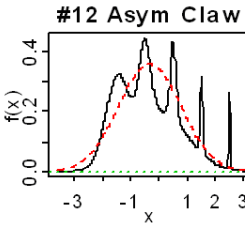
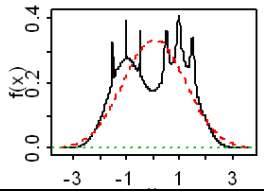
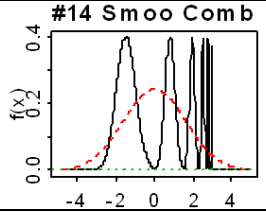
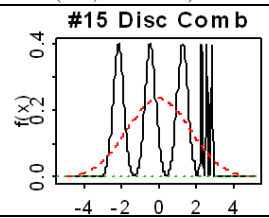
```
c=distribution('Exponential(10)');
d=distribution('Max(Exponential(10);Normal(10;1))');
c() % returns a value that follows the distribution c
c(1); d(1); % resets seed number generator to 1, to allow re-generate the
   same values later
for k=1:100; p(k)=c(); end; % put 100 values in p, following the
   distribution c
for k=1:100; q(k)=d(); end; % put 100 values in q, following the
   distribution d
```

[7] http://www.mathworks.com/

Appendix II

The distributions Marron-Wand are defined using probabilities of returning different Normal distributions with different means and variances. In each line we of the definition, we put the probability, followed by the mean and variance. Note that in Distribution View format the arguments of the Normal distribution are the mean and the standard deviation.

Marron-Wand function densities and definitions	
	
1; Normal(0;1)	0.2; Normal(-0.3; 1.4400000) 0.2; Normal(0.3; 0.6400000) 0.6; Normal(1.0; 0.4444444)
	
0.125; Normal(0.000000; 1.000000000) 0.125; Normal(-1.000000; 0.444444444) 0.125; Normal(-1.666667; 0.197530864) 0.125; Normal(-2.111111; 0.087791495) 0.125; Normal(-2.407407; 0.039018442) 0.125; Normal(-2.604938; 0.017341530) 0.125; Normal(-2.736626; 0.007707347) 0.125; Normal(-2.824417; 0.003425487)	0.6666667; Normal(0; 1.00) 0.3333333; Normal(0; 0.01)
	
0.1; Normal(0; 1.00) 0.9; Normal(0; 0.01)	0.5; Normal(-1; 0.4444444) 0.5; Normal(1; 0.4444444)
	
0.5; Normal(-1.5; 0.25) 0.5; Normal(1.5; 0.25)	0.75; Normal(1.0000000; 0.0) 0.25; Normal(0.1111111; 1.5)

#9 Trimodal  	#10 Claw 
0.45; Normal(-1.2; 0.3600) 0.45; Normal(1.2; 0.3600) 0.10; Normal(0.0; 0.0625)	0.5; Normal(0.0; 1.00) 0.1; Normal(-1.0; 0.01) 0.1; Normal(-0.5; 0.01) 0.1; Normal(0.0; 0.01) 0.1; Normal(0.5; 0.01) 0.1; Normal(1.0; 0.01)
#11 Doub Claw  	#12 Asym Claw 
0.490000000; Normal(-1.0; 0.4444444) 0.490000000; Normal(1.0; 0.4444444) 0.002857143; Normal(-1.5; 0.0001000) 0.002857143; Normal(-1.0; 0.0001000) 0.002857143; Normal(-0.5; 0.0001000) 0.002857143; Normal(0.0; 0.0001000) 0.002857143; Normal(0.5; 0.0001000) 0.002857143; Normal(1.0; 0.0001000) 0.002857143; Normal(1.5; 0.0001000)	0.50000000; Normal(0.0; 1.000000) 0.25806452; Normal(-1.5; 0.160000) 0.12903226; Normal(-0.5; 0.040000) 0.06451613; Normal(0.5; 0.010000) 0.03225806; Normal(1.5; 0.002500) 0.01612903; Normal(2.5; 0.000625)
#13 As Do Claw  	#14 Smoo Comb 
0.460000000; Normal(-1.0; 0.4444444) 0.460000000; Normal(1.0; 0.4444444) 0.003333333; Normal(-1.5; 0.0001000) 0.003333333; Normal(-1.0; 0.0001000) 0.003333333; Normal(-0.5; 0.0001000) 0.023333333; Normal(0.5; 0.0049000) 0.023333333; Normal(1.0; 0.0049000) 0.023333333; Normal(1.5; 0.0049000)	0.50793651; Normal(-1.4761905; 0.2579994961) 0.25396825; Normal(0.8095238; 0.0644998740) 0.12698413; Normal(1.9523810; 0.0161249685) 0.06349206; Normal(2.5238095; 0.0040312421) 0.03174603; Normal(2.8095238; 0.0010078105) 0.01587302; Normal(2.9523810; 0.0002519526)
#15 Disc Comb  	
0.28571429; Normal(-2.1428571; 0.081632653) 0.28571429; Normal(-0.4285714; 0.081632653) 0.28571429; Normal(1.2857143; 0.081632653) 0.04761905; Normal(2.2857143; 0.002267574) 0.04761905; Normal(2.5714286; 0.002267574) 0.04761905; Normal(2.8571429; 0.002267574)	