2D Geometric Model of N-Dimensional Lotteries in Python

James Yu

November 12, 2020

Abstract

A lottery is a set of probabilities over a series of outcomes. These probabilities create a linear combination of the outcomes to given an expected value. Given the set of all possible probabilities, it is possible to construct a geometric representation of a particular set of lotteries over a set of outcomes. We construct a recursive model for representing a set of N-dimensional lotteries as a dynamic 2-dimensional graph using Python and PyGame. The result can be found at https://repl.it/@JPhoton/ExpectedUtilityVisualizer

Introduction

In the lecture notes for UBCV ECON 304 provided on November 5th, the concept of a lottery was introduced (Li, Lecture Notes for expected utility_20_1). Lotteries are sets of probabilities over outcomes which form a linear combination that provides an expected value. For example, consider outcome space $C = (1, 2, 3)$ and lottery $L = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ over C. This lottery has expected value:

$$Ex[L] = \frac{1}{3} + \frac{2 * 1}{3} + \frac{3 * 1}{3} = \frac{6}{3} = 2 \tag{1}$$

This represents equal chances of every outcome. We could instead have lottery $L' = (1, 0, 0)$ over C with expected value:

$$Ex[L] = 1 * 1 + 0 * 2 + 0 * 3 = 1 \tag{2}$$

which represents a 100% chance of the first outcome.

Also provided in the lecture notes was a geometric representation of the set of all possible lotteries over an outcome set. For a three outcome case, the representation is a triangle with the outcomes at each vertex. Each vertex represents a certain chance of getting that outcome, edges represent linear combinations of the vertices attached to them, and the interior points represent linear combinations of the three outcomes. These three conditions are crucial to the model.

The model works for $N = 3$ outcomes, but becomes more challenging when N is greater than 3. The goal of this paper is to demonstrate a method of representing these higher-dimensional lottery sets.

Procedure

The process of achieving the model was lengthy and complex. Initially, the thought was that since the $N = 3$ case was a 3-vertex polygon, a general N-outcome case would be an N-vertex polygon such that the same three conditions applied as before, and with the midpoint representing an equal probability for all outcomes.

Thus, a rendering setup was constructed with Python 3 using the PyGame user interface framework. An engine was written to take a set of outcomes and represent them as an N-vertex polygon with the outcomes at the vertices.

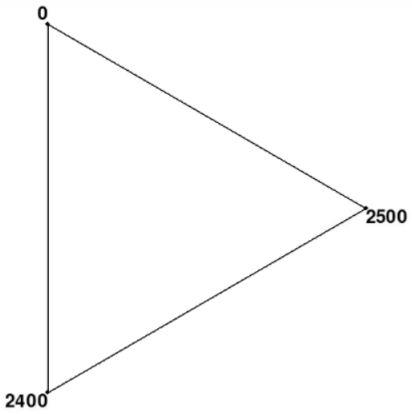


Figure 1. Rendering of a three-outcome model.

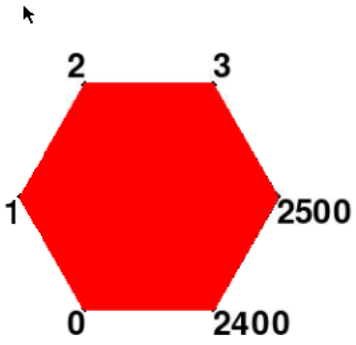


Figure 2. Rendering of a six-outcome model, shaded red to test future indifference curve displaying.

Upon completing this model, an issue arose. Given that the midpoint represents an even distribution of probabilities, and that a corner point represents a 100% chance of that particular outcome, the probability of said outcome ranges from $\frac{1}{N}$ to 1 along half the length of the diagonal of the polygon, and $\frac{1}{N}$ to 0 along the other half, for number of outcomes N.

In other words, given a line, the distribution of probabilities over the line is nonlinear unless $N = 2$. This required a method of obtaining variable density on a line.

An iterative approach was used. Consider $N = 4$. The midpoint of the line has probability 0.25, one endpoint 0, and one endpoint 1. On a normal line, the midpoint would be 0.5, so a function $f : [0, 1] \to [0, 1]$ was necessary to map the values. Said function would need to preserve $f(0) = 0$ and $f(1) = 1$ where $f(0.5) = 0.25$.

The solution to this is $f = x^2$. $0^2 = 0$, $1^2 = 1$ and $0.5^2 = 0.25$, as needed. This procedure also describes $N = 8$, for example, where $\frac{1}{8} = 0.125$ and $0.5^3 = 0.125$ giving the solution to this scenario as $f = x^3$.

Thus, given N outcomes each with probability $\frac{1}{N}$, the exponent x of the polynomial x^k that fits the variable density line solves the equation:

$$0.5^k = \frac{1}{N} \tag{3}$$

Solving for k, we get $0.5^k = N^{-1}$, so:

$$k = -1 * \frac{log_2(N)}{log_2(0.5)} \tag{4}$$

We choose base 2 since $log_2(0.5) = -1$, which allows us to say:

$$k = log_2(N) \tag{5}$$

meaning an N-outcome lottery would have a variable density line defined by:

$$f(x) = x^{log_2(N)} \tag{6}$$

Upon completion of this representation, however, another issue occurred. While in the case where N is even, we would have a diagonal along which a variable density line could be applied, no such diagonal exists in the odd case. This would have raised significant difficulty if not for the following two realizations:

First, variable density lines may represent the probability of one particular outcome, but the points in the model are meant to represent multiple outcomes at once and so the line between them need not have variable density.

Second, and more importantly, this model is fundamentally flawed at higher dimensions. Consider the $N = 4$ case with a 4-vertex polygon, or a square. This would look somewhat like the below:

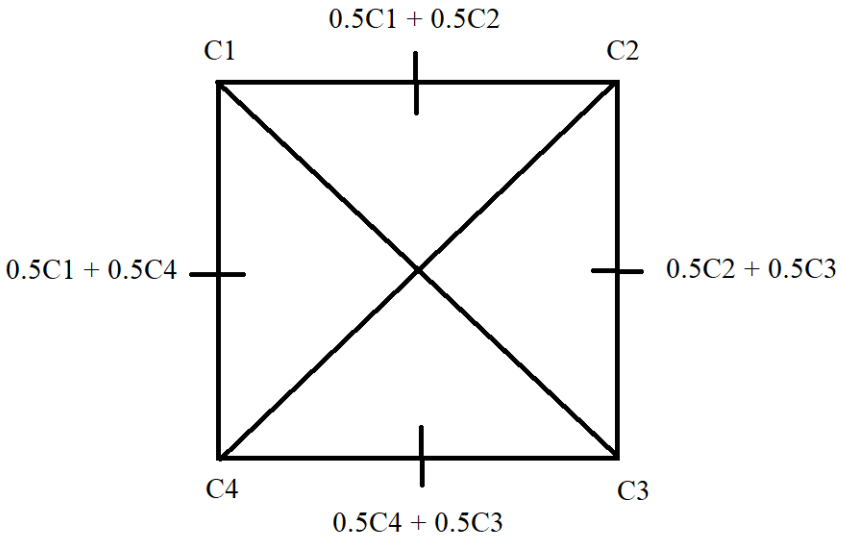


Figure 3. A sample $N = 4$ polygon model.

At first, the model seems fine. A relation can be made between outcomes along the edges, with the equal distribution case in the middle. However, one quickly realizes that there are two obvious lotteries that the model fails to represent: $L = 0.5C_1 + 0.5C_3$ and $L' = 0.5C_2 + 0.5C_4$. Using the diagonals, one would think those might lie in the center. However, they cannot both lie in the center as they may have different expected values. Additionally, the center already represents $0.25C_1 + 0.25C_2 + 0.25C_3 + 0.25C_4$. It is true that this lottery is the composition $L'' = 0.5L + 0.5L'$, but representing lotteries in this form loses the ability to specifically pinpoint L and L'.

Thus, a modified model would be necessary. The polygon model was abandoned due to lack of ability to represent all outcomes, but not completely ignored. The square case was used to derive a method of generalizing the triangle representation from before.

Consider an $N = 3$ case. We know a triangle can represent this case, and each point represents a particular linear combination.

Add an extra outcome to achieve $N = 4$. For every lottery that we had before, we can now either choose one of those lotteries, or we can choose a lottery of $(0, 0, 0, 1)$, or some linear combination of the two, creating a compound lottery.

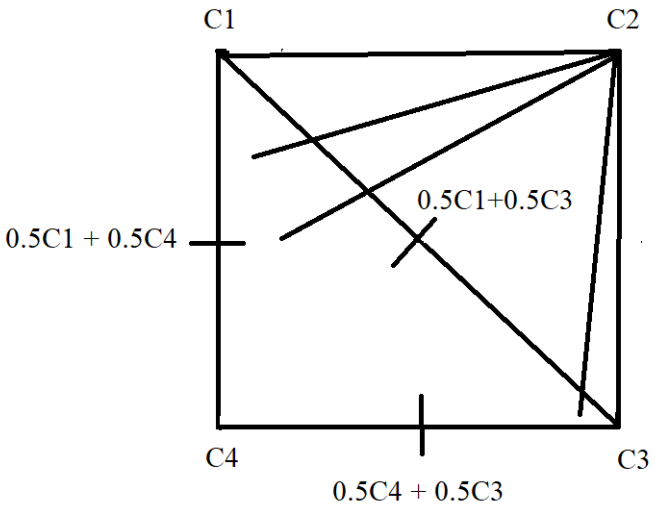


Figure 4. $N = 4$ polygon model from before modified to show linear combinations of $N = 3$ and C_2.

The above case generalizes to a triangular pyramid. Thus, consider the following game tree for $N = 3$.

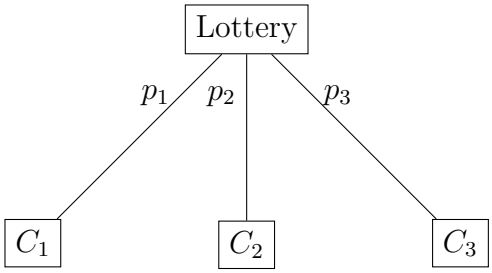


Figure 5. $N = 3$ game tree in the general case.

Let $p_1 + p_2 + p_3 = 1$ for $p_1, p_2, p_3 \in [0, 1]$. Now, we add an additional outcome C_4 with probability $p_4 = 1$ and create a compound lottery between a chance $q_2 \in [0, 1]$ of C_4 and a chance $q_1 \in [0, 1]$ of the previous lottery.

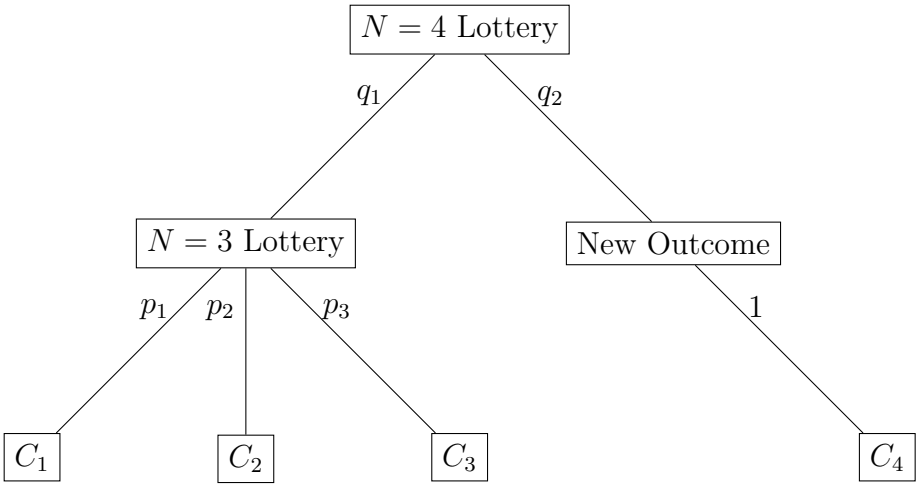


Figure 6. $N = 4$ compound lottery game tree.

Let $q_1 + q_2 = 1$. We can compute the reduced lottery L_R of this compound lottery as:

$$L_R = (q_1p_1, q_1p_2, q_1p_3, q_2) \text{ over } (C_1, C_2, C_3, C_4) \tag{7}$$

Since $p_1 + p_2 + p_3 = 1$ and $q_1 + q_2 = 1$, we see the sum of the outcomes of L_R is:

$$sum(L_R) = q_1p_1 + q_1p_2 + q_1p_3 + q_2 = q_1(p_1 + p_2 + p_3) = q_1 + q_2 = 1 \tag{8}$$

as needed. This gives us a simple lottery over $N = 4$ outcomes. We can then choose values of p_1, p_2, p_3, q_1 and q_2 to cover all the possible cases.

We extend this model to the general k-ary lottery and prove this works by induction. Consider the base case where $k = 1$, which gives a single point with probability $p = 1$.

Our inductive step is that given a k-ary compound lottery L which has a reduced form, we can construct a $k + 1$-ary compound lottery L'' which also has a reduced form. Let the inductive hypothesis be that L exists. We have:

$$L = (p_1, ..., p_k) \text{ over } (C_1, ...C_k) \tag{9}$$

where $p_1, ..., p_k$ are the probabilities of each of the k outcomes in the reduced lottery of L. Extend L to $(p_1, ..., p_k, 0)$ to account for the new outcome later.

Let $L' = (0, ..., 0, 1)$ be the degenerate lottery of a new outcome C_{k+1} for a $k+1$-outcome lottery. Construct the compound lottery L'' as:

$$L'' = q_1 L + q_2 L' \tag{10}$$

for $q_1, q_2 \in [0, 1]$ where $q_1 + q_2 = 1$. We can then compute the reduced lottery as:

$$L''_R = (q_1 p_1, ..., q_1 p_k, 0) + (0, ..., 0, q_2) = (q_1 p_1, ..., q_1 p_k, q_2) \tag{11}$$

which is a $k+1$-ary reduced form of compound lottery L'', as was to be shown. QED.

Thus, we have a recursive model for representing a lottery of N outcomes as a series of compound lotteries linked to each other. We can now represent this in two dimensional space.

Let the base case be $N = 2$ to fit the two dimensions we are working with. This gives us a line of probabilities between two outcomes C_1, C_2 ranging from 0 to 1 where 0 is a 100% chance of C_1, 0.5 is a half chance of both, and 1 is a 100% chance of C_2. Unlike the variable-density line from equation (6), we can use a standard line to represent this as it is the basic 2-dimensional case.

Let p_1 be a point on that line, and let $p_2 = 1 - p_1$. Consider outcome C_3 and link it to p_1 with another line of values from 0 to 1. Consider point q_2 on this new line. We have the probability of C_1 as $p_2 q_2$, C_2 as $p_1 q_2$, and C_3 as $1 - q_2$, which precisely represents the recursive model we constructed.

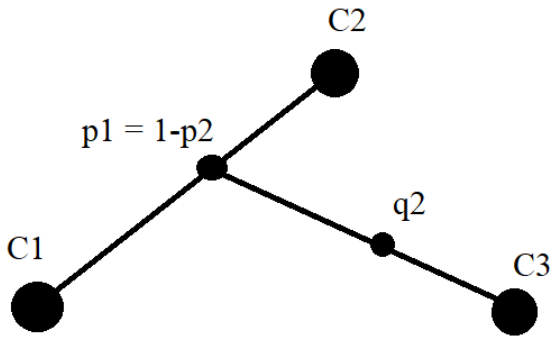


Figure 7. Geometric representation of $N = 3$ outcomes using the recursive model.

By varying p_1 and q_2, we are able to create a range of points which together, by connecting each of C_1, C_2, C_3 together, forms a triangle as in our original example. Because our model is recursive, and because the representation directly represents the model, we are able to extend this representation to a general N-ary lottery.

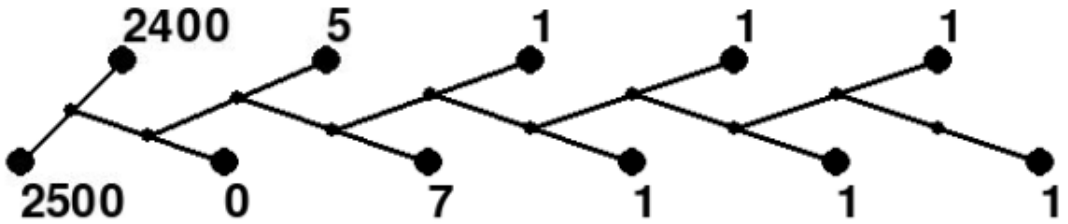


Figure 8. Geometric representation of $N = 11$ outcomes using the recursive model.

Now that the geometric model exists, we create the computer program to render it. We begin by rendering the outcomes onscreen as in Figure 8: approximately half of the outcomes are below the middle of the screen, and half are above.

We draw a line between the first two outcomes to represent the base case, and then draw a point on that line at an arbitrary position to represent the starting p_1. From there, we draw a line from p_1 to the next outcome and draw a point at an arbitrary q_2. At first, these points were halfway along the line by default, but were later changed to be set up such that every outcome initially had an equal chance of occurrence.

We then connect a line from q_2 to the next outcome and repeat the process until we arrive at the Nth outcome. We draw a final arbitrary point on the line connecting it to the previous arbitrary point, and end the algorithm here.

After ironing out the issues with the rendering system and constructing a way to move the sliders based on mouse input, a method was necessary to convert from coordinates to probabilities and back. As the probabilities are defined as fractions of lines, it was necessary to determine the fraction of a line a probability point slider was located at.

By basic geometry and similar triangles, if we have two vertices of a line at (x_0, y_0) and (x_2, y_2), and a slider is positioned at (x_1, y_1), the fraction p of a line can be given by:

$$1 - p = \frac{x_2 - x_0}{x_2 - x_1} = \frac{y_2 - y_0}{y_2 - y_1} \tag{12}$$

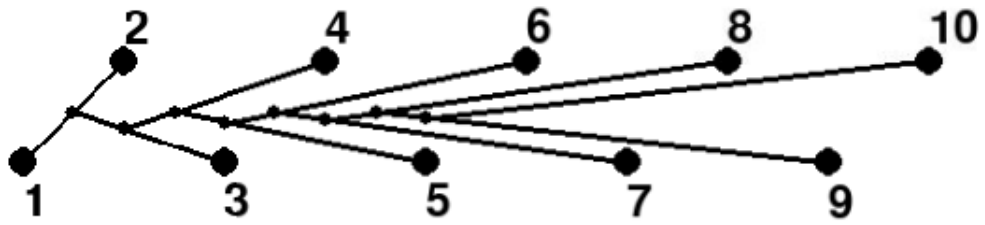
where $1 - p$ is the fraction of the line to the right of a particular probability slider.

Now that probabilities can be obtained, expected value can be computed. First, we calculate the actual probabilities recursively by the recursive model we developed: given a set of base probabilities $p_1, ..., p_k$, each further outcome results in the prior probabilities being multiplied by $(1 - p_{k+1})$ where p_{k+1} is the fraction defined by the $(k + 1)$th probability slider. We then insert p_{k+1} into the list of probabilities and keep going.

Given these probabilities, we compute:

$$Ex[L] = r_1C_1 + ... + r_NC_N \tag{13}$$

for probabilities $r_1, ..., r_N$ of the reduced N-ary lottery L being represented, and outcomes $C_1, ..., C_N$. This gives us expected value.



Expected Value: 0.1*1 + 0.1*2 + 0.1*3 + 0.1*4 + 0.1*5 + 0.1*6 + 0.1*7 + 0.1*8 + 0.1*9 + 0.1*10 = 5.5

Figure 9. Geometric representation of $N = 10$ with uniform probabilities and expected value.

Following this, we compute variance as:

$$Var(L) = p_1(C_1 - Ex[L])^2 + ... + p_N(C_N - Ex[L])^2 \tag{14}$$

where $Ex[L]$ is the expected value from (13).

Given these metrics, we now have a fully computerized 2-dimensional representation of any N-ary set of lotteries over a set of N outcomes. We now add additional features to the UI of the program for adding and removing outcomes.

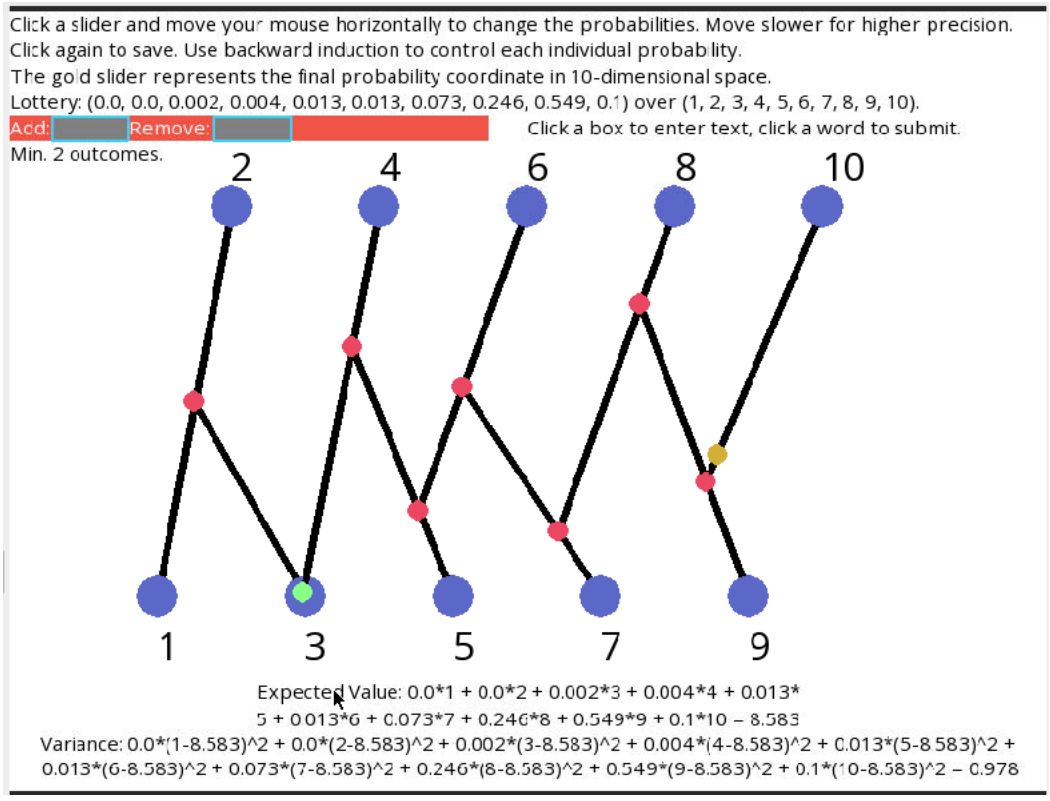


Figure 10. Work in progress program with add and remove buttons.

With more of the UI finished, the ability to modify the utility function was added. The utility function is a function $u : \mathbb{R} \to \mathbb{R}$ which maps outcome values to a personal level of utility. Computing the expectation of this result with respect to the probabilities gives expected utility, which itself has its own variance.

Finally, indifference fields were added to the program. Indifference fields are sets of parallel $(N-1)$-dimensional hyperplanes of an N dimensional representation of an $(N+1)$-ary lottery such that the expected utility along every point of the hyperplane is the same.

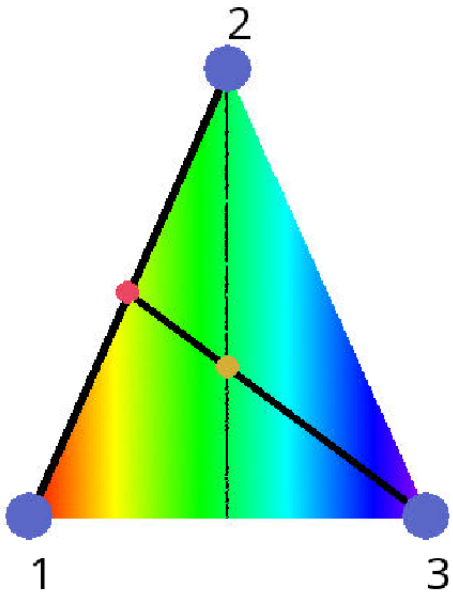


Figure 11. Work in progress program with indifference fields for $N = 3$ outcomes and $u(x) = x$.

To make visibility clearer, the indifference fields are represented as bands of color. The expected utility is mapped from a scale of minimum to maximum value to a scale from 0 to 276 and converted to an HSL color with 50% luminance and 100% saturation. This gives a range of values from red to purple with the bluer values representing higher utility. Later in development when risk aversion was factored in, the scale ended up falling outside of these bounds, such that the color bands wrapped from purple to pink to back to red.

To actually compute the indifference fields, the program recursively iterates over every possible combination of probabilities. As this is computationally intensive at higher dimensions, the density by which probabilities are selected i.e. the amount of values between 0 and 1 was intentionally reduced at higher dimensions. A slider was later added to control this density to provide a greater range of display options.

After extending the indifference field rendering algorithm to multiple dimensions, the final step was to factor in risk aversion. Risk aversion represents an individual's propensity to seek or avoid a riskier situation i.e. a lottery with higher variance. Risk aversion can be divided into three categories:

Risk averse, where an individual seeks to avoid risk;

Risk neutral, where an individual disregards risk and rather takes regard to expected utility only;

Risk seeking, where an individual seeks out riskier lotteries intentionally.

Thus, we see that the final preferences an individual takes over a set of lotteries is partly dependent on expected utility and partly dependent on variance. At the most extreme cases, the individual either has utility positively related to variance or negatively related, and in the middle case, the individual has utility positively related to expected value. We thus define the indifference field utility function for lottery L as:

$$v(x) = (1 - |r|) * Ex[L] - sign(r) * |r| * var(L) \tag{15}$$

for $r \in [-1, 1]$, $sign(r \geq 0) = 1$ and $sign(r < 0) = -1$.

The completely risk averse case is given by $r = 1$ where $1 - |r| = 1 - 1 = 0$ and $sign(r) * |r| = 1$, which reduces $v(x)$ to $-var(L)$, as needed.

The risk neutral case is given by $r = 0$ where $1 - |r| = 1 - 0 = 1$ and $sign(r) * |r| = 1 * 0 = 0$, which reduces $v(x)$ to $Ex[L]$, as needed.

Finally, the risk seeking case is given by $r = -1$ where $1 - |r| = 1 - |-1| = 1 - 1 = 0$ and $sign(r) * |r| = -1 * |-1| = -1 * 1 = -1$, which reduces $v(x) = var(L)$, as needed.

Intermediate cases of r thus give a linear combination of $Ex[L]$ and $var(L)$. The magnitude of r represents the weight of variance relative to expected utility, and the sign of r represents whether the individual leans toward or away from risk.

This concludes the construction of the program, the result of which can be found at https://repl.it/@JPhoton/ExpectedUtilityVisualizer online.

Observations and Discussions

Numerous observations were conducted over the course of the development of the program, the first of which was made even before the program was written. While the model used in this program is recursively based on individual compound lotteries, another model was found that accomplishes the same effect with a different method of recursion. In this alternative recursion model, lotteries are compounded in groups of 1 or 2 instead of one at a time, forming an H-like structure.

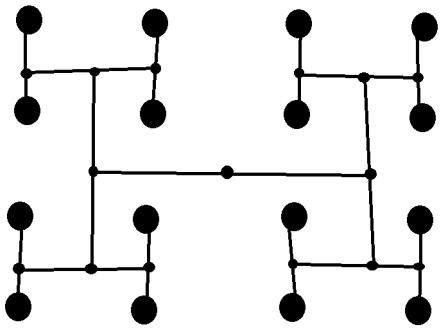


Figure 12. H-based recursive lottery model, not pursued due to its recursive complexity.

This structure can be redrawn into a different form: that of a binary tree, with the leaves being the outcomes and the parent nodes' values being the probability sliders.

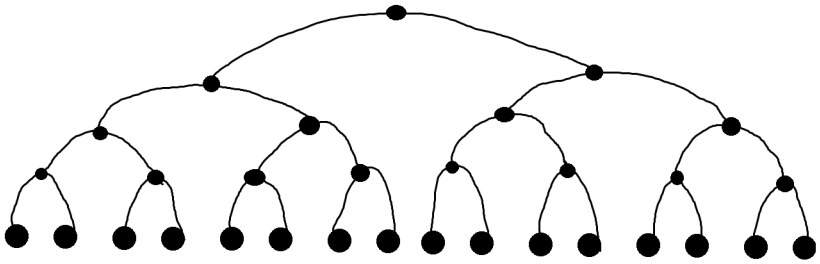


Figure 13. Tree-based recursive lottery model where the probability sliders may be radial dials.

These models were not pursued further due to their complexity in terms of both recursion and rendering, but nonetheless may be of interest for further study. The binary tree representation in particular, being a model well-used in computer science, has the array of tools used to study binary trees accessible to it, including the possibility of constructing a binary search tree where the probability sliders in the tree are sorted values.

Next, an observation was noted regarding convergence of compounded probabilities. Given a reduced lottery, the sequence of probabilities which define the additions at each compounding step, in the case where each final probability is the same, geometrically converge to a straight line.

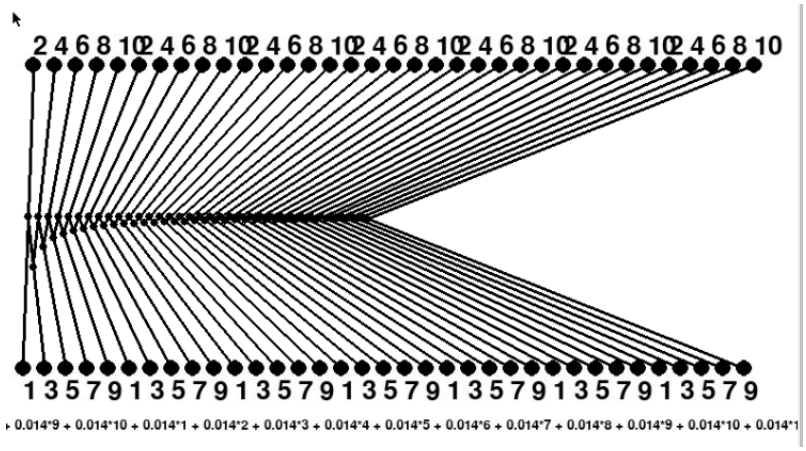


Figure 14. Convergence of probabilities observed at an intermediate level of outcomes.

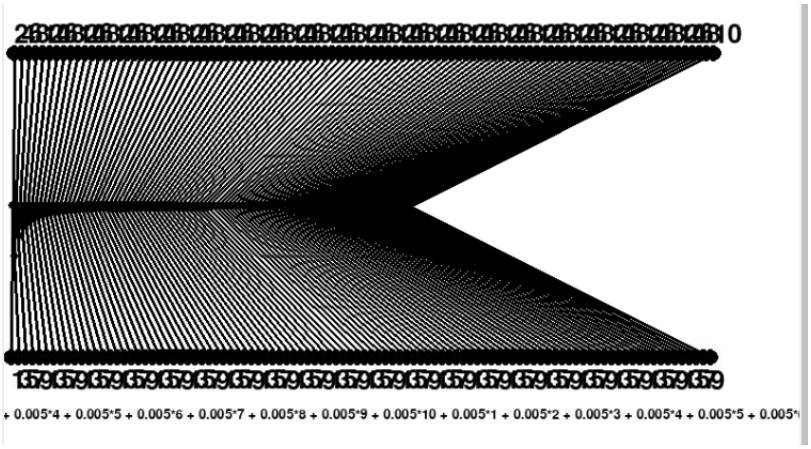


Figure 15. Convergence of probabilities observed at a high level of outcomes.

This appears to be associated with the line fraction probabilities themselves converging closer and closer to zero, which as the number of outcomes goes to infinity, averages to a geometric convergence line.

Additionally, given a sufficiently large amount of outcomes, adjusting one of the leftmost probability sliders results in a well-defined curve which propagates to all subsequent sliders.

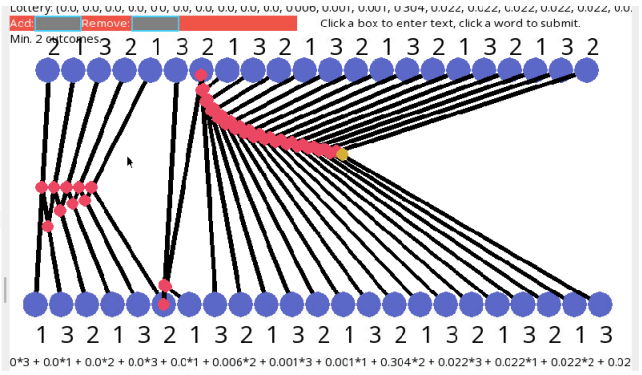


Figure 16. Convergence of probabilities with initial bias.

The next observations conducted have to do with the form of the indifference fields. Initially, the indifference fields at higher dimensions were computed with 100% density, meaning they would appear as continuous sheets of color. However, the indifference fields were no longer always straight nor parallel, which was initially mysterious. The field was also not aligned to the actual final slider.

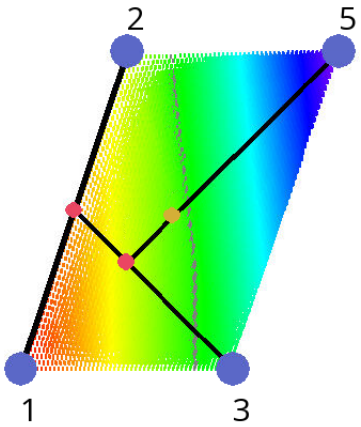


Figure 17. $N = 4$ outcomes with a skewed indifference field.

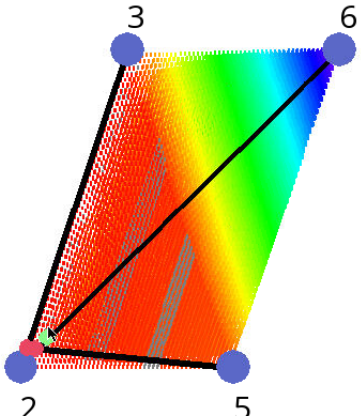


Figure 18. $N = 4$ outcomes for $u(x) = x^x$ with a two-dimensional indifference field.

Upon further investigation, it turned out that rendering N-dimensional fields at $N > 2$ with 100% density was a mistake. Doing so obscured the fact that the indifference fields were actually not always lines, which was the initial belief. Rather, they are $(N-1)$-dimensional hyperplanes.

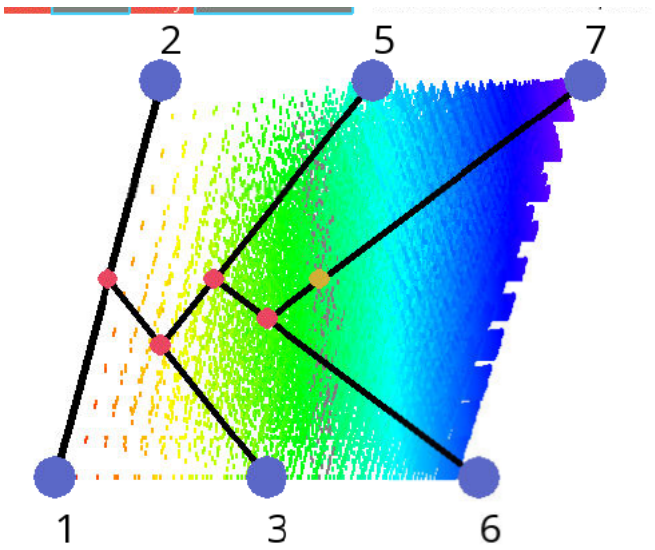


Figure 19. A 5-dimensional structure unveiled by not running at 100% density.

It then became apparent that the program was capable of physically rendering representations of N-dimensional shapes instead of just rendering a 2-dimensional wave of color. These shapes are in fact N-dimensional hyperpyramids with $(N-1)$-dimensional recursive hyperpyramid bases as defined by the recursive model.

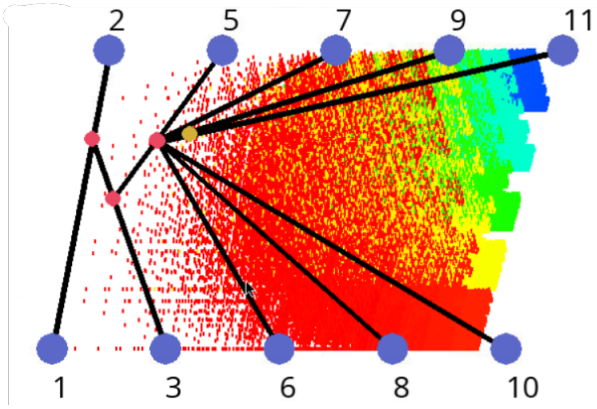


Figure 20. 9-dimensional set of indifference fields for $u(x) = x^x$.

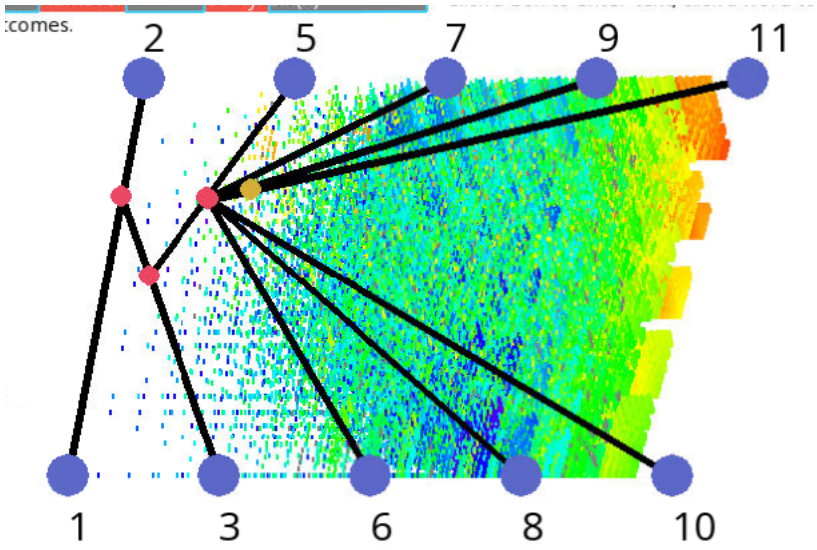


Figure 21. 9-dimensional set of fields for $u(x) = sin(x)$. The hyperplanes are clearly visible.

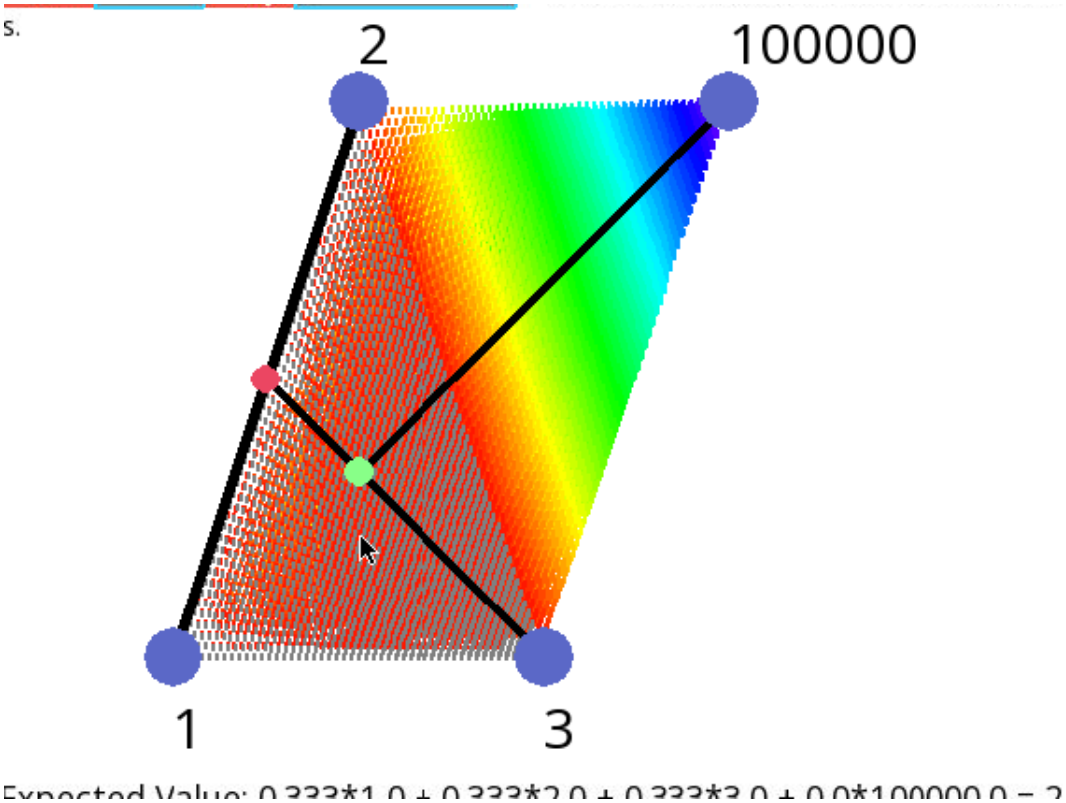


Figure 22. 3-dimensional set of planar fields which form a triangular pyramid.

Experimentation was then conducted on lotteries where multiple outcomes have the same value. The result was indifference fields which tended to extend across the model.

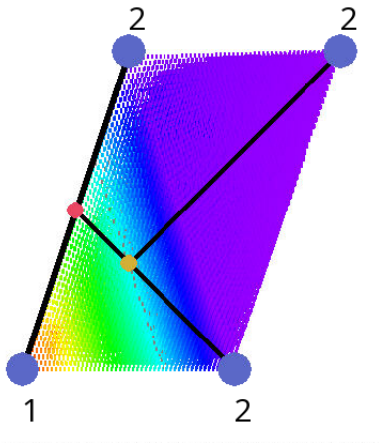


Figure 23. 3-dimensional field set with repeated entries.

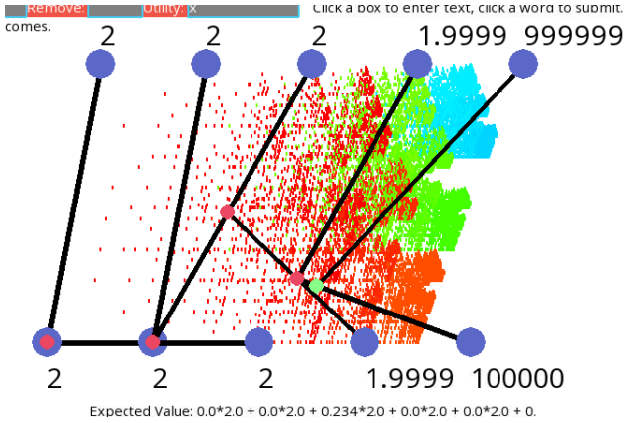


Figure 24. 9-dimensional field set with repeated entries.

Final experimentation was conducted with risk aversion applied. Upon doing so, what was originally a set of indifference hyperplanes was quickly invalidated to reveal indifference parabloids.

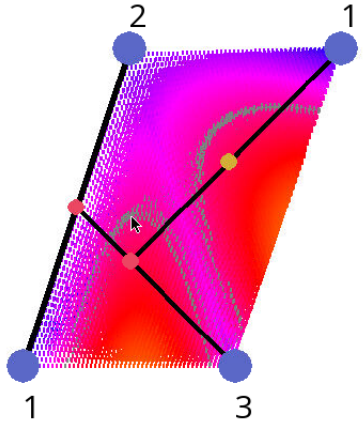


Figure 25. Early test with risk-seeking nature revealing parabloid fields.

An alternate method of representing risk aversion by taking the reciprocal instead of the negative of the variance was tested, but found to be insufficient to represent utility due to the rapidity at which it would cycle through the HSL color scale.

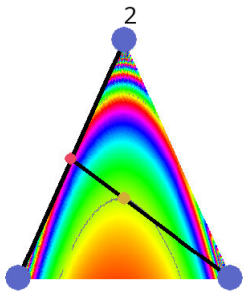


Figure 26. Paraboloid indifference fields in 2 dimensions.

The current risk aversion function, minus the risk-seeking component, was then applied.

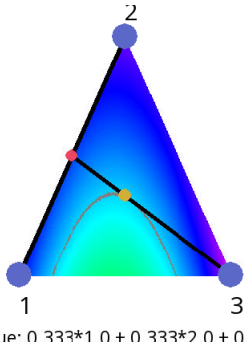


Figure 27. More paraboloid indifference fields in 2 dimensions. Highest utility is at the low-risk edges.

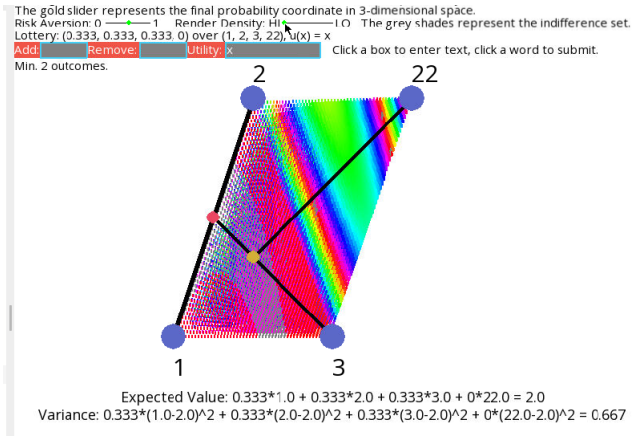


Figure 28. 3 dimensional indifference field set with 50% risk aversion.

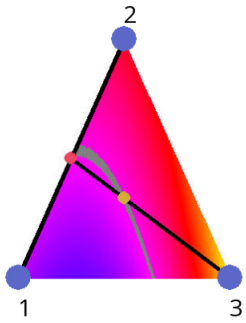


Figure 29. 2 dimensional indifference field set with partial risk aversion, showing partial parabloids.

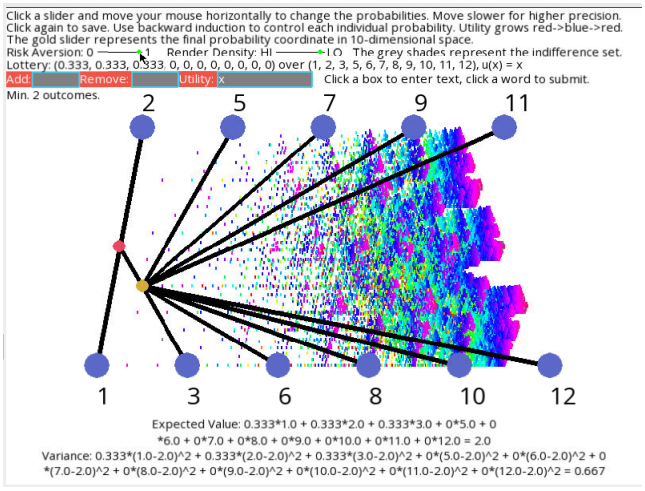


Figure 30. 10-dimensional indifference field set with high risk aversion.

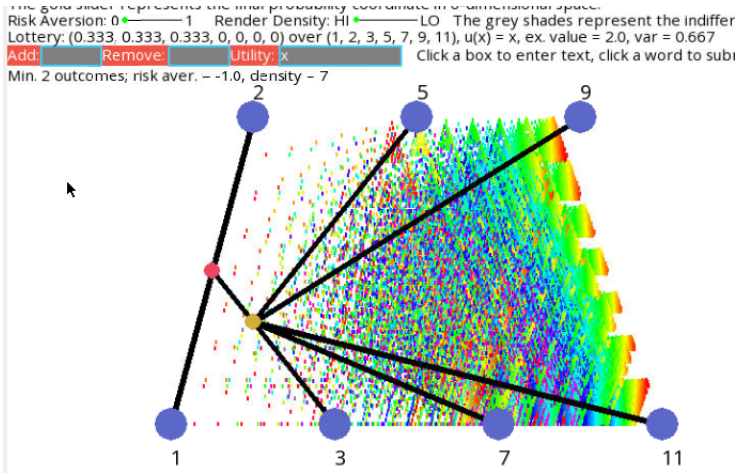


Figure 31. 6-dimensional field set with risk seeking nature. The zero slider marking was an error.

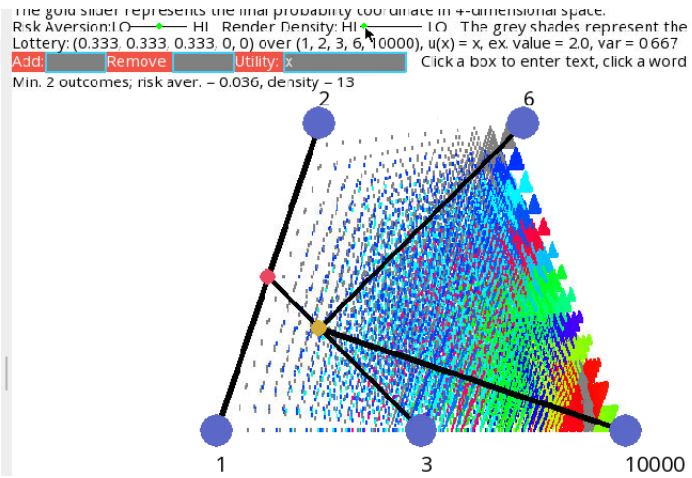


Figure 32. 4-dimensional field set with near-neutral risk aversion showing indifference triangular pyramids.

Conclusion

A recursive compound lottery model of representing an N-dimensional lottery in 2-dimensional space is successful at demonstrating the range of possible lottery outcomes. The model is able to represent every linear combination of probabilities possible, indifference fields, and can adapt to risk aversion. Beyond lotteries and economics, the model may be useful for those seeking a way to geometrically represent higher dimensional objects in 2-dimensional space, and for those seeking a line-based user interface for controlling data.

Works Cited

Li, W. *Expected Utility.* Personal collection of W. Li, University of British Columbia, Vancouver, BC.