

# The Many Worlds of Random Portfolios

---



**Patrick Burns**  
**<http://www.burns-stat.com>**

**February 2008**

---

This talk and the other talks of the evening focus on the technical issue of generating random portfolios. We neither know how to generate random portfolios exactly from a given distribution, nor do we have consensus on what distribution we actually want to generate. Given this, there is a grave danger – in my opinion – of thinking that random portfolios are not currently useful. As Jason MacQueen commented, all of quantitative finance is approximation. Random portfolios are an incredibly powerful tool with many applications. They are very useful even though we can't generate them precisely.

A list of some applications (and of references) is on the random portfolios page of the Burns Statistics website.

# CONSTRAINT

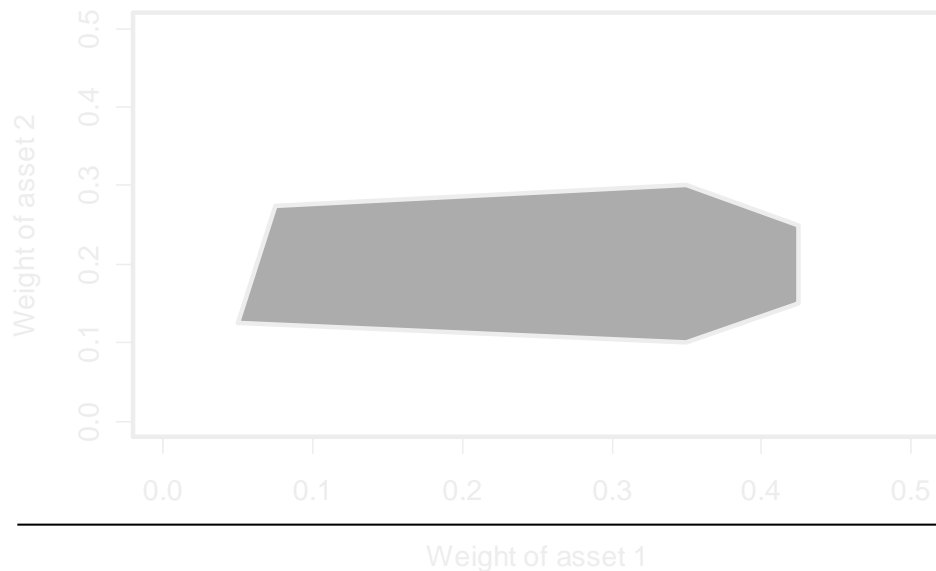
Constraints are the key to random portfolios. It is constraints that define the random portfolios – constraints provide the boundaries.

Linear constraints, like sector or country constraints, are the easiest to deal with mathematically.

Quadratic constraints, like a volatility constraint, are harder to deal with.

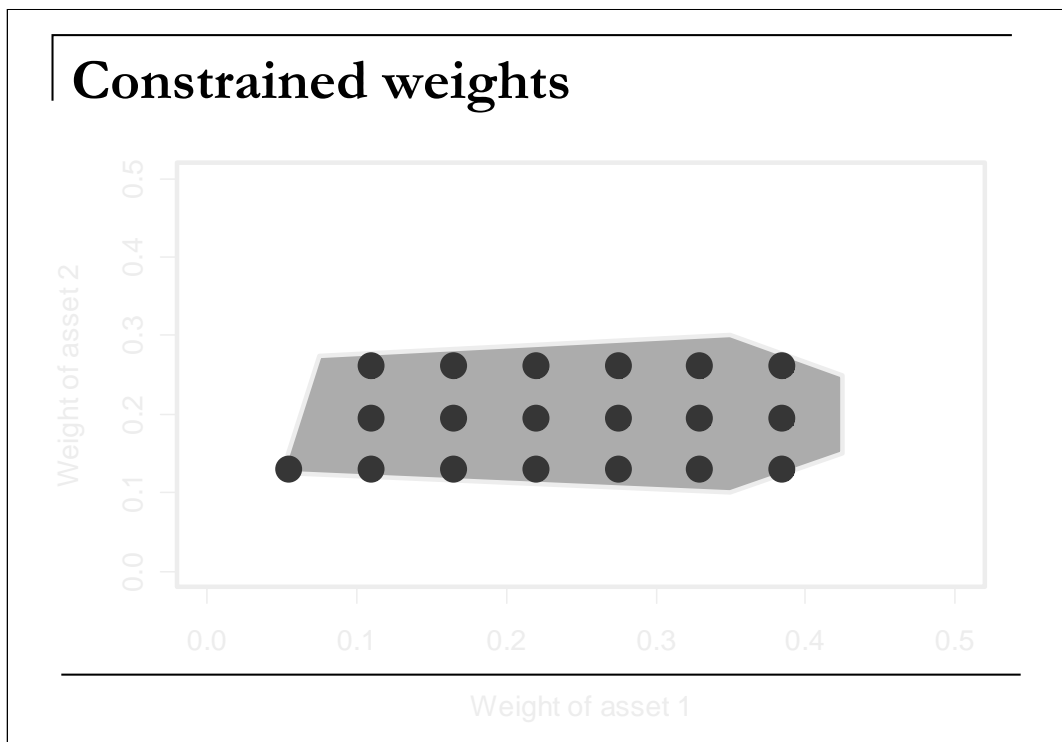
Integer constraints, like the maximum number of names to hold or to trade, are horrendously difficult mathematically.

## Constrained weights



The coloured portion of the plot shows the weight combinations of two assets in a three asset problem that are allowed within our constraints.

The first distribution that comes to mind for generating random portfolios is the uniform distribution. If weights can be any number between zero and one, then we need measure theory to understand the uniform distribution. The probability that you want to go there is extremely low – I don't, and I have a PhD in statistics.



But if there is only a finite number of possible portfolios within our constraints, then the uniform distribution is much easier to understand. In fact there is only a finite number of possible portfolios. Suppose that we are building portfolios from constituents of the FTSE 100 and the amount of money in the portfolio is between £10,000,000 and £10,000,100. Since we can't buy fractional shares, there is a finite number of possible portfolios we can construct.

With a finite number of possible portfolios, generating from the uniform distribution means that each portfolio inside the constraints has an equal chance of being selected. If we generate 100 times as many portfolios as possible portfolios, then each portfolio will appear just about 100 times if we are generating from the uniform.

In the example, the smallest allowed weight for asset 1 appears only a third as much as the other allowed weights. There is no simple way to tell if we are generating from the uniform distribution. In particular, the weights of each asset are not uniform.

## **Universe 0**

### **Uniform distribution**

The game is that we have decided on the constraints that we want. With each decision beyond that we will be thrown into a different universe.

Our first decision is to generate from the uniform distribution. We now have another choice: exactly uniform or approximately uniform.

## Universe 1

### **Uniform distribution -- exact**

This universe is on hold for practical sized problems (and even for ridiculously small problems). We need to find a 17 year old genius from somewhere to tell us how to do this.

If we could generate uniformly with no constraints (except the particular universe of assets), then there is a way to generate exactly uniformly with constraints: reject any portfolio that is generated that does not satisfy the constraints. The problem here is that with practical constraints you will reject almost all portfolios. It wouldn't be surprising to need to wait centuries between accepted portfolios.

## Universe 2

### Uniform distribution -- approximate

We **can** generate from approximately uniform.

There are a mess of universes that branch off from here depending on how we measure approximate.

The real way to tell if we are uniform is to generate several times as many portfolios as there are possible portfolios inside the constraints and then see if each possible portfolio is represented about the right number of times. When the number of possible portfolios is larger than the number of protons in the universe, then we have a problem.

We need an approximate way to tell how approximate we are to uniform. This seems like an easier problem than generating exactly from the uniform, but another 17 year old genius might not hurt.

## A Wedding in Cherokee County

<u>Parents</u>		<u>Raw Twins</u>		<u>Twins</u>	
A .2	D .2	A .2	D .4	A .222	D .364
D .4	G .3	D .2	E .1	D .222	E .091
E .1	P .3	P .3	G .3	P .333	G .273
Z .3	Z .2	Z .2	Z .3	Z .222	Z .273

### Interlude: Genetic Algorithms

A key use of genetic algorithms is for optimisation. Most optimisation algorithms hold a single candidate solution at a time. Genetic algorithms are different – they hold a population of solutions at each point in time.

In our case a “solution” is a portfolio. One approach to the genetics is illustrated in the slide. Two parents create a set of twins. The assets in both parents are in both twins, the other assets are randomly assigned. The weights are tied to the asset but otherwise randomly assigned. Then the weights need to be scaled in the twins.

This is much better than the original genetic algorithm for this problem. The original genetic algorithm is astoundingly bad, for several reasons.



We can formulate the generation of random portfolios as an optimisation with a random algorithm, such as a genetic algorithm.

The function we want to minimise is zero if all constraints are met, and it is positive when constraints are broken – the bigger the violations, the bigger the function value.

In the image the lake is the flat spot at elevation zero. The lake is completely surrounded by hills where the constraints are violated. The genetic algorithm lands at a random spot on the hill, it kicks a rock and then sees where the rock splashes into the lake.

Just like real rocks, the rocks in our virtual lake will tend to land near the shore: once we satisfy all the constraints we quit.

**Universe 161518207514**

**Found distribution**

Now we have a new distribution – the “found” distribution. That is, whatever distribution our algorithm gives us. In the case of the genetic algorithm we just saw, we expect constraints to be nearly binding a lot more than with the uniform distribution.

Using the found distribution in our applications seems like a stupid thing to do. Maybe it is a stupid thing to do, but maybe not. We’ll come back to that.

**Universe 161518207514 + n**

**Found distribution + n random steps  
(approximately uniform)**

There is a clever idea (unfortunately not mine) to go from a found distribution to approximately uniform. Once we are in the lake (that is, have a solution that meets all the constraints), then take some number of additional steps that leave us still in the lake. The more steps we take the more likely we are to be anywhere in the lake.

David Jessop wondered how many steps it would take to get to approximately uniform. That will obviously depend on the step sizes. It will also undoubtedly depend on the dimensionality of the lake. And of course it will depend on your definition of approximate.

## Universe 42

### Exact fund manager distribution

Let's think about a third distribution: the fund manager distribution. The fund manager will satisfy all the constraints but it is hard to believe that they will be uniform inside the constraints. They are likely to be close to binding on some constraints. If the fund manager has sector constraints, they are likely to be near the maximum of their currently favourite sector.

Thus the fund manager distribution MIGHT look something like the found distribution that we just saw.

Generating from the exact fund manager distribution seems hopelessly out of reach, but I claim we already have the technology to do that if the fund manager is doing an optimisation (with alphas). Standardise the alphas by the asset volatilities, randomly permute the standardised alphas (randomly switch the names), do an optimisation with the permuted (rescaled) alphas. The downside is speed: doing a full portfolio optimisation for just one random portfolio.

Universe  $e^{-\pi i}$

**Found distribution = fund manager  
distribution**

Suppose by some magic that the found distribution exactly matches the fund manager distribution. Should we use the found distribution in this case?

There is Richard Young's philosophy that you always want to use the uniform distribution. But there is an argument not to use it even if your philosophy is to match the fund manager. We often have to approximate the constraints. Using a distribution with extra mass near the edges makes the analysis more sensitive to the particular approximation we select. By this argument you want to move at least part way towards uniform unless you know the constraints exactly.

There is a counter argument that moving towards the uniform biases the analysis, and only gives the illusion of less sensitivity.

<b>Multiverse</b>		
	<b>Exact</b>	<b>Approx.</b>
<b>Uniform</b>	<b>too hard</b>	<b>attainable</b>
<b>Found</b>	<b>attainable</b>	
<b>Fund manager</b>	<b>sometimes (slow)</b>	<b>like found?</b>

Not all fund manager distributions are the same, not even for the same fund manager. Each found distribution is different – seemingly small changes to the algorithm can change the distribution.

The best choice of distribution will depend on the application. But almost certainly any choice is better than no choice. Random portfolios can and should change the way that fund management is conducted, even in the absence of perfect generation techniques.